Developpez Magazine

Edition de Décembre-Janvier 2007/2008.
Numéro 13.
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

Le magazine Developpez.com est de retour pour cette nouvelle année.

Découvrez dans ce nouveau magazine, une série d'articles, de critiques de livres, de questions/réponses sur diverses technologies.

La rédaction

Article PHP

Tests et benchmark en PHP 5

Cet article vous montrera les différences de performances entre différentes manières de faire en PHP.

par **Matthieu Fernandez**
Page 9

Article PHP

Haute disponibilité avec MS-SQL Server

Découvrez les différentes manières de sécuriser son serveur MS-SQL Server pour garantir la plus haute disponibilité possible pour ses bases de données.

par **Frédéric Brouard**
Page 39

JTables - Un autre regard

Ce tutoriel a pour but de comprendre le fonctionnement des JTables en utilisant un système d'explication distancié du pattern Modèle/Vue/Controlleur sans toutefois contester la validité de ce pattern. Il s'adresse aux débutants en Swing ou aux programmeurs plus expérimentés n'ayant utilisé principalement que les fonctions par défaut des JTables.

1. Introduction

Les JTables sont des composants Java écrits par Sun Microsystem pour gérer les tableaux. Sun fournit des Objets par défaut permettant de les utiliser très simplement comme tableur. Mais il est également possible de mettre ce que l'on veut dans chacune des cases (même un autre JTable!) et d'aller plus loin qu'un tableur classique.

Vous trouverez plus bas du code tiré directement d'Edupassion.com, site web proposant entre autres des bulletins de note. Parfois le code est coloré en rouge, permettant de retrouver un même objet entre différents objets.

2. Ce que l'utilisateur voit

2.1. L'utilisateur voit des cellules : les TableCellRenderer

Si ce que l'on voit est un texte ou un nombre, on aura sans doute affaire à un new DefaultTableCellRenderer() qui hérite de JLabel. Chaque case peut etre différente : c'est la fonction TableCellRenderer.getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) qui definira le JComponent selon la ligne et la colonne. Si getTableCellRendererComponent renvoit un JButton, l'utilisateur pourra cliquer dessus. Si le JComponent est un JLabel, on peut lire une donnée.

Voici le code pour obtenir le JComponent de type JButton dans lequel apparaît le nom de l'élève.

```
public class RendererEleve extends javax.swing.JButton
implements javax.swing.table.TableCellRenderer {

    public RendererEleve() { /*Constructeur vide*/   }

/** l'object value représente en principe l'élève */

    public java.awt.Component
getTableCellRendererComponent(javax.swing.JTable table,
Object value, boolean isSelected, boolean hasFocus, int
row, int col)
    {
        String nom=((Eleve) value).toString();// Prenom
+ Nom de l'élève
        this.setText(nom);
        return this;
    }
}// Fin de la classe
```

2.2. L'utilisateur lit des données dans ces cellules

Ces données sont stockées dans le TableModel. La fonction TableModel.getValueAt(int row, int column) renvoie l'objet métier selon la colonne et la ligne.

```
public class TableModel {

    public Object getValueAt(int row, int column) {

        if
(column==this.INDEX_COLONNE_LISTE_DES_ELEVES){ //
Colonne des élèves
            Eleve eleve =
(Eleve)listeDesEleves.get(row);
            return eleve;
        }

        if (column==this.INDEX_COLONNE_MOYENNE){
//Colonne des moyennes
             Eleve eleve =
(Eleve)listeDesEleves.get(row);
            return trouveMoyenne(eleve);
          }
    }
}
```

L'objet renvoyé sera traité par la fonction TableCellRenderer.getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) vue auparavant (I.A).

2.3. L'utilisateur comprend que les cellules sont rangées par colonne

La plupart du temps, les éléments d'une colonne ont la même signification : une date, un nom, un prénom, et ici les notes. Les fonctionnalités d'une colonne sont gérées par une TableColumn.

```
public class TableColumnEleve extends TableColumn{
    /**
     * Crée un style pour les TableColumn des Eleves,
et spécifie le modelIndex à 0.
     */
    public TableColumnEleve() {
        super(0); // le modelIndex est à 0, ce qui
indique que les élèves seront représentés à la
première colone
        setHeaderRenderer(new HeaderEleve());//Le
component sera un HeaderEleve, qui hérite d'un JLabel
        setCellRenderer(new RendererEleve());//Le
Component est un RendererEleve qui hérite d'un JButton
```

```
    }
}// Fin de TableColumnEleve
```

Ces TableColumn sont ensuite réparties lors de la création du JTable

```
public class Bulletin extends JTable {
       BulletinModel model;

       public Bulletin(Cours cours) {
          /* Création du modèle */
        this.setAutoCreateColumnsFromModel(false);
        BulletinModel model= new BulletinModel(cours,
this); //Le modèle dépendra des notes dans le Cours
        this.setModel(model);

        /* Création des TableColumn */
        createColumns();
        }

       private void createColumns() {
           /* Premiere TableColumn : celle contenant la
liste des Eleves */
            TableColumnEleve tableColumnEleve = new
TableColumnEleve();
            this.addColumn(tableColumnEleve);

            /* Ajout des controles : On a une nouvelle
colonne par controle donné aux élèves */
            for (int
modelIndex=model.INDEX_PREMIER_CONTROLE;modelIndex <=
model.INDEX_DERNIER_CONTROLE;modelIndex++){
                Controle controle= (Controle)
listeDesControles.get(modelIndex-
model.INDEX_PREMIER_CONTROLE);
                TableColumnControle columnControle=new
TableColumnControle(modelIndex,
controle);//Contrairement à TableColumnEleve , l'index
est dans le constructeur
                this.addColumn(columnControle);
            }
                    //On continue avec les autres
colonnes
       }

}//Fin de la classe Bulletin
```

2.4. La colonne comprend un entête

L'entête est de la classe HeaderRenderer : il s'agit encore d'une TableCellRenderer qui par défaut est grisâtre (selon le JRE utilisé). Si votre table est grande, vous la ferez scroller vers le bas, mais l'entête reste. C'est pourquoi l'entête n'est pas géré graphiquement par le container du JTable, mais par le scrollPane parent.

```
this.scrollPane.getViewport().add(this.bulletin);//Perme
t au scollpane de gérer le header - bulletin derive de
JTable
```

Chaque colonne contient un Header, qui est donc choisi dans le TableColumn

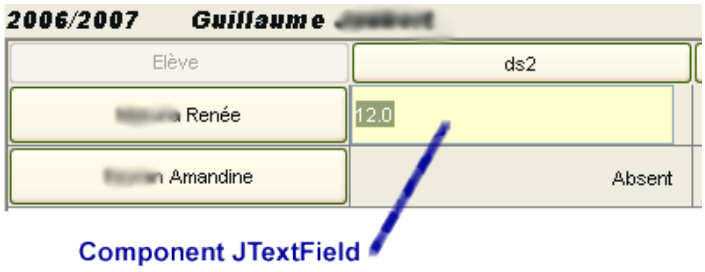
```
public class TableColumnEleve extends TableColumn{
    /**
     * Crée un style pour les TableColumn des Eleves, et
spécifie le modelIndex à 0.
     */
    public TableColumnEleve() {
        super(0); // le modelIndex est à 0, ce qui
indique la première colone
        setHeaderRenderer(new HeaderEleve());//Le
component sera un HeaderEleve, qui gérite d'un JLabel
disabled
        setCellRenderer(new RendererEleve());//Le
Component est un RendererEleve qui hérite d'un JButton
    }
}// Fin de TableColumnEleve
```

3. Ce que l'utilisateur fait

3.1. L'utilisateur va cliquer sur une case pour en modifier le contenu

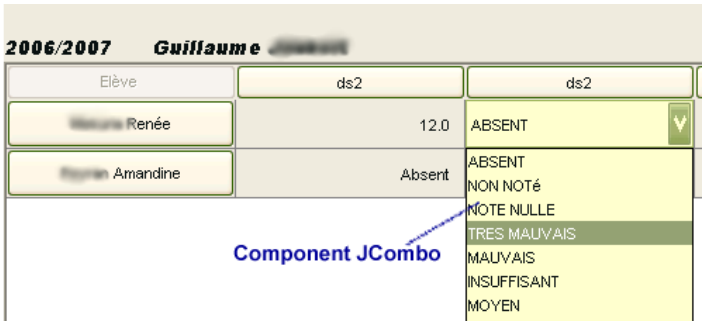
Si la case est éditable, alors on rentre dans le mode d'édition de la cellule, géré par un objet de classe TableCellEditor. Il apparaît à l'écran un nouvel objet : le CellEditorComponent. Il s'agit d'un JComponent classique tel un JTextField ou un JCombo.

Style de note Européenne :NoteEUEditor



La valeur "12.0" est surlignée : NoteEUEditor reçoit l'objet value puis le programmeur écrit du code pour afficher cet objet à sa façon. On a choisit ici de présélectionner le texte, mais on pourrait par exemple afficher value/2. Le malheureux élève aurait sa note divisée par deux chaque fois qu'un professeur clique sur cette case.

Style de note Primaire (NotePrimaireEditor)



On obtient le CellEditorComponent d'édition dans **TableCellEditor : public Component getTableCellEditorComponent(JTable table, Object value, boolean isSelected, int row, int column)**

```
public class NoteEUEditor extends AbstractCellEditor
implements TableCellEditor{

    public NoteEU noteEU;
    JTextField fieldNoteEU=null; // C'est le
JComponent que l'utilisateur va voir quand il entrera
dans le mode Editor.
    JTable table;

    /** Constructeur pour NoteEUEditor
     * Cet éditeur permet de modifier une note sur /20
avec un seul clic.
```

```
     */
    public NoteEUEditor() {
        this.fieldNoteEU=new JTextField ();
    }

    /**
     * Cette fonction permet l'affichage sous forme de
String de la note de l'élève
     * L'objet value est référencé dans le
TableModel.getValueAt().
     */
     public Component getTableCellEditorComponent(JTable
table, Object value, boolean isSelected, int row, int
column)
     {
            NoteEU noteEU=(NoteEU) value; //L'objet
Value est ce qui apparait AVANT que l'on modifie la
valeur - ici c'est une note sur 20
            String stringNote=
((Float)noteEU.getValue()).toString();//StringNote est
la chaine de caractère que l'on va afficher dans le
JTextField

            /* Traitement conditionnel selon les
spécificités métier */
            if (noteEU.getValue()<-2000)
fieldNoteEU.setText("");//Cas pour un élève absent,
représenté par la note -10002
            else labelNoteEU.setText(stringNote);


                    /* Une fois que tous les calculs
sont faits, on sélectionnera les chiffres de la note */
            SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                fieldNoteEU.requestFocus();
                fieldNoteEU.selectAll();// utilisé pour
que tout soit sélectionné, mais ca marche pas...
              }
               });

            return fieldNoteEU; //Le JTextField contient
maintenant ce que l'on veut.
     }
}
```

3.2. L'utilisateur fait une modification

Si on valide le changement, on renvoie un objet grace à la fonction public Object getCellEditorValue() dans le même TableCellEditor.

```
public class NoteEUEditor extends AbstractCellEditor
implements TableCellEditor{

       (... Voir plus haut ...)

      /**
       * Renvoie un objet une fois APRES MODIFICATION par
l'utilisateur
       * L'objet renvoyé est très basique et sera ensuite
traité par TableModel.setValueAt().
       */
     public Object getCellEditorValue()
     {
         String str=labelNoteEU.getText();
         if (str.equalsIgnoreCase("ABS")||
str.equalsIgnoreCase("a")||
str.equalsIgnoreCase("absent")) return
Note.FLOAT_ABSENT; //On renvoit -10002
         if (str.equalsIgnoreCase("")) return
Note.FLOAT_NON_NOTE;
         try{
             Float f=new Float(str);
             if
(f.floatValue()<=Note.FLOAT_NA.floatValue()) return
Note.FLOAT_NA;
             else return f;
         }
         catch (Exception e)
{e.printStackTrace();return Note.FLOAT_NA;}//Si
l'utilisateur a rentré n'importe quoi (ce qui arrive
souvent ;) )
     }
}
```

Arrive alors un processus invisible pour l'utilisateur grâce aux différents Events préprogrammés par nos amis de Sun Microsystem. L'objet renvoyé par AbstractCellEditor : public Object getCellEditorValue() arrive à TableModel.setValueAt(Object obj, int row, int col), ce que nous détaillerons plus bas.

Il est vivement conseillé de passer par AbstractCellEditor afin de ne pas reprogrammer des méthodes comme fireEditingStopped(). En effet l'AbstractCellEditor fera souvent la démarche logique souhaitée.

3.3. Le Model est modifié

Les observateurs auront notés que la fonction getCellEditorValue() ne fait pas référence à la position de la cellule dans le tableau. Cela a pour avantage de pouvoir réutiliser le TableCellEditor dans d'autre JTable (voire JTree ou Jlist avec une utilisation mineure de l'héritage). L'influence de la position dans le tableau dépend logiquement du traitement métier, et est traitée dans le TabelModel.setValueAt().

```
public void setValueAt(Object obj, int row, int col) {
        if (col==this.INDEX_COLONNE_APPRECIATION)
this.setValueForAppreciation(obj, row);
        if (col==this.INDEX_COLONNE_MOYENNE)
this.setValueForMoyenne(obj,row,col);
                else
                        this.setValueForControle(obj,
row, col);
    }
```

L'Editor renvoie un Objet, et en fonction de celui-ci, le TableModel modifie ses données. Les systèmes d'événement programmés par Sun mettront à jour la partie visuelle en faisant appel à TableModel.getValueAt() avec les nouvelles valeurs.

4. Ce que JTable fait... et ne fait pas

4.1. Une couche de verre

Lorsque l'on clique sur un bouton de la JTable, on ne clique pas sur le bouton, mais sur la JTable, qui transmet ensuite l'événement au bouton... si on l'a programmé. On dit parfois qu'il y a une glace de verre par dessus la Table : il est possible de voir ce qu'il y a en dessous, mais en appuyant à un endroit, on appuie sur toute la table.

4.2. Un exemple classique : Cliquer sur un bouton du Header, ce qui nous permettra de rajouter un Controle à notre Bulletin

On va créer un MouseListener qui va écouter ce que notre souris clique.

```
public class EvtAjoutEtModifDeControle implements
MouseListener{

    Bulletin bulletin;
    JTableHeader header;
```

```
    /**
     * Creates a new instance of
EvtAjoutEtModifDeControle
     */
    public EvtAjoutEtModifDeControle(Bulletin bulletin)
{
        this.bulletin=bulletin;
    }

    public void mouseClicked(MouseEvent e) {

       /* Ligne clé ! c'est ici qu'on sait où on clique
*/
        int indexDeColonneSelected =
bulletin.convertColumnIndexToModel(bulletin.columnAtPoin
t(e.getPoint()));

               /* code une fois que l'on a su où l'on
cliquait */
        if
(indexDeColonneSelected==bulletin.getBulletinModel().IND
EX_COLONNE_AJOUT_DEVOIR)
            MaFactory.creerControle(bulletin);
        }

   /* Les autres méthodes non prises en charge */
    public void mousePressed(MouseEvent e) {    }
    public void mouseReleased(MouseEvent e) {    }
    public void mouseEntered(MouseEvent e) {    }
    public void mouseExited(MouseEvent e) {    }
}
```

Et dans le constructeur de la JTable, on rajoute la ligne :

```
this.getTableHeader().addMouseListener(new
EvtAjoutEtModifDeControle(this));
```

Pour continuer avec notre allégorie, il faut ordonner à chaque JComponent de "regarder" ce qui se passe au dessus de la glace posée sur la Table.

4.3. Une conséquence : Utiliser autant que possible les objets et méthodes par défaut, afin de ne pas devoir reprogrammer les événements.

J'ai donné tantôt un exemple de JComboBox fourni lorsque l'on clique sur une cellule. On voudrait qu'il se passe ceci : User choisit dans le combo un objet_choisit -> public void monJComboActionPerformed() -> public Object monEditor.getCellEditorValue() -> monModel.setValueAt (objet_choisit , row, col)

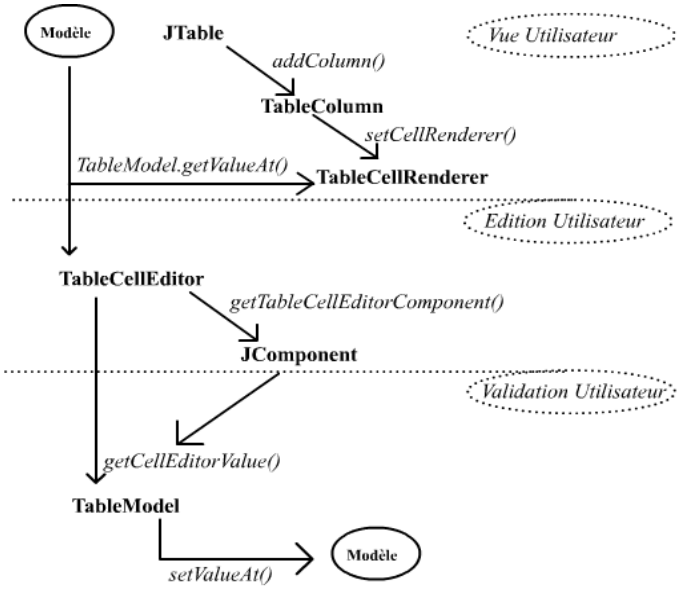
Malheureusement, une fois que l'on choisit l'élément de notre JComboBox, la fonction programmée JComboBoxActionPerformed() ne s'exécute pas : l'événement n'est pas transmis, et il faudrait reprogrammer les enchaînement des événements. C'est faisable, mais pénible, c'est pourquoi Sun Microsystem nous donne un DefaultCellEditor (JComboBox combo) - il est possible d'avoir un DefaultCellEditor basé sur d'autres JComponents.

Le DefaultCellEditor s'occupe des événements, et il vous reste à faire le code métier. Dans notre cas, la dernière ligne des bulletins est occupée par les moyennes du Controle et il n'y a donc pas de choix possible : voyons la nouvelle implémentation pour l'éditeur de notes type Ecole Primaire.

```
public class NotePrimaireEditor extends
DefaultCellEditor{

    /** Creates a new instance of NoteUSEditor */
    public NotePrimaireEditor(JComboBox combo) {
        super (combo);
    }

   public Component getTableCellEditorComponent(JTable
table, Object value, boolean isSelected, int row, int
column) {

        /* Si c'est la dernière ligne, on affiche la
moyenne */
        if (row ==
((BulletinModel)table.getModel()).INDEX_DERNIERE_LIGNE
){
            if (value!=null)
                return new JLabel(value.toString());
            else return new JLabel();
        }
        else //On peut afficher le JComboBox
            return
super.getTableCellEditorComponent(table, value,
isSelected, row, column);


    }


    /** La fonction est à implémenter si vous voulez
renvoyer un autre objet que celui présent dans le
JComboBox (par exemble à partir d'une Factory).
     * Sinon, c'est le DefaultCellEditor qui renvoie
l'objet contenu dans le JCombo. C'est le traitement
par défaut car c'est le plus logique.
    *public Object getCellEditorValue() {}
    */
}
```

5. Conclusion

5.1. Récapitulatif



Les objets de La VUE UTILISATEUR définissent ce que l'utilisateur voit avant d'intéragir avec le JTable. Le TableCellRenderer dessine sur l'écran selon les données du Model,les directives données par le TableColumn, puis l'index de ligne ou de colonne.

Les objets de l'EDITION UTILISATEUR définissent en général le composant interactif (comme un formulaire HTML)

qu'utilisera l'utilisateur quand il aura cliqué sur la cellule.

Enfin, la VALIDATION UTILISATEUR se fait en deux étapes. A partir de l'interaction utilisateur, on définit un objet à partir des fonction getCellEditorValue (qui ne dépend pas de l'index ligne/colonne) et/ou de setValueAt (qui dépend de l'index ligne/colonne) .

5.2. Et quand j'aurai tout compris à ce tutoriel ?

Voici de quoi faire pas mal de chose avec les JTables. Cependant, les JTables étant des composants Swing, les bonnes pratiques de Swing s'appliquent aussi, notamment une bonne gestion des Threads (cf. SwingUtilities en section II-A).

Si votre programme a des accès longs vers la base de données, vous devrez utiliser d'autres fonctions des JTables, comme EditCellAt().
Si votre programme est complexe (volontairement ou non), vous devrez peut-être redéfinir les Events, ou réécrire les fonctions fireXXX() des objets fournis par Sun.

Mon avis est que l'utilisation des JTables est déjà assez compliquée, et si vous souhaitez que votre code soit maintenable par une autre personne, évitez au maximum de réécrire ce que Sun Microsystem a déjà fait. Concrètement, cela consiste à écrire des classes qui héritent des DefaultXXX ou AbstractXXX plutôt que d'implémenter à nouveau les Interfaces.

Retrouvez l'article de Nicolas Zozol en ligne : Lien1

Interview des auteurs de Spring par la pratique

1. Vous, votre travail et vos publications

Bonjour à vous, et un grand merci d'avoir accepté cette interview. Tout d'abord, pour les personnes qui ne vous connaissent pas, pouvez vous vous présenter en quelques mots ?

Thierry Templier :
Bonjour Gildas. C'est avec plaisir pour cette interview!
Je suis architecte applicatif spécialisé dans les technologies objet et Web au sein de la société Argia Engineering dont l'activité consiste en de l'architecture, de l'expertise sur les technologies Java/java EE et Web ainsi que de l'assemblage de solutions open source avec une très forte part de modélisation. Je travaille sur ces technologies depuis 8 ans et utilise au quotidien les différents outils et technologies abordés dans le livre, à savoir Spring bien sûr, mais également Hibernate, DWR, GWT, XFire, JMS, JMX...
D'un autre côté, je suis également le co-auteur du livre "JavaScript pour le Web 2.0" aux éditions Eyrolles, ouvrage décrivant les différentes facettes du langage JavaScript et des bibliothèques de ce type afin d'implémenter des applications Web à l'interface graphique riche.

Julien Dubois :
Bonjour et merci également de nous interviewer.
Cela fait maintenant 10 ans que je travaille dans le développement informatique, dont les 8 dernières années à ne faire que du Java/J2EE. Actuellement je dirige une équipe de 25 personnes, chez un éditeur de logiciel français. Nous sommes spécialisés dans le traitement de larges volumes de données et la très grande majorité de nos développements est réalisée avec les technologies décrites dans "Spring par la pratique" : Spring 2.0, Hibernate, ActiveMQ, DWR, Struts...

Jean-Philippe Retaillé :
Bonjour Gildas. Je suis architecte technique au sein d'un grand groupe d'assurance européen depuis plusieurs années. J'interviens essentiellement sur des projets Web J2EE avec des problématiques d'intégration fortes. Je suis également l'auteur de "Refactoring des applications Java/J2EE" et un des co-auteurs de "Programmation orientée aspect pour Java/J2EE", tous deux chez Eyrolles.

Quelles ont été les motivations qui vous ont poussé à écrire Spring par la pratique ?

Thierry Templier :
L'écriture de Spring par la pratique a été motivé par l'envie de faire partager mon enthousiasme pour ce framework. Nous avions également envie d'adresser des problématiques d'architecture d'applications Java EE ainsi que des bonnes pratiques de développement. Ma rencontre avec Jean-Philippe et Julien a permis de concrétiser ce projet.

Julien Dubois :
Une telle occasion de passer toutes mes nuits et mes week-end devant mon PC, je ne pouvais pas refuser :-)
Plus sérieusement, j'avais déjà écrit plusieurs articles et cela m'avait énormément plu, alors faire un livre était la suite logique. Je souhaitais également faire partager mon expérience, et je trouvais très intéressant de le faire avec Thierry et Jean-Philippe, qui sont deux personnes que j'apprécie beaucoup.

Jean-Philippe Retaillé :
A l'origine, nous avions l'intention d'écrire un livre sur les bonnes pratiques Java/J2EE car nous trouvions qu'il y avait un manque dans ce domaine parmi les ouvrages français. Après avoir rédigé quelques chapitres, nous nous sommes rapidement aperçus que Spring était très souvent cité et utilisé pour implémenter nos exemples de bonnes pratiques. Nous avons donc décidé de réorienter le livre exlusivement sur Spring et de le centrer sur une étude de cas complète afin de nous différencier des approches adoptées par les ouvrages anglo-saxons disponibles sur Spring à l'époque.

Comment vous êtes vous connus tous les trois ?

Thierry Templier :
J'ai fait la connaissance de Julien par l'intermédiaire de mon ancienne société Sogeti en 2004. Nous nous intéressions à des sujets communs dont Spring... En parallèle, j'étais en contact avec Jean-Philippe qui m'était entré en relation avec moi pour son livre "Refactoring des applications Java/J2EE" afin d'utiliser une application open source que j'avais développée.

Julien Dubois :
Thierry et moi avons travaillé en même temps chez Capgemini (en 2004), on s'est rencontrés là-bas. A l'époque, peu de monde s'intéressait à Spring, et j'ai un collègue qui m'a donné le nom de Thierry.
C'est Thierry qui m'a ensuite présenté à Jean-Philippe, qu'il connaissait déjà pour avoir travaillé avec lui sur son livre "Refactoring des applications Java/J2EE".

Jean-Philippe Retaillé :
Tout a commencé avec mon livre "Refactoring des applications Java/J2EE". Je cherchais une étude de cas sous forme d'un projet open source existant ni trop complexe, ni trop simpliste et de préférence français. Je suis alors tombé sur JGenea qui a été développé par Thierry (que je ne connaissais pas à l'époque). J'ai donc pris contact avec lui pour obtenir l'autorisation d'utiliser son application et Thierry s'est proposé d'être relecteur technique de mon livre. Cela a suffit pour lui inoculer le virus de l'écriture. Je lui ai tout naturellement proposé de l'accompagner pour son premier ouvrage. D'expérience, je préfère travailler à 3 sur ce genre de projet et Thierry a immédiatement pensé à Julien.

Avez-vous d'autres publications prévues ?

Thierry Templier :
Quelques articles pour developpez.com autour des technologies OSGi, Spring et JavaScript!

Julien Dubois :
Peut-être un article pour Oracle Technology Network d'ici quelques mois.

2. Vous et Spring Framework

Comment avez vous connu Spring, et qu'est-ce qui vous a poussé à l'utiliser ?

Thierry Templier :
C'est en 2004 que j'ai commencé à m'intéresser au framework. Nous étions en train de regarder l'impact des tests unitaires dans les applications, ce qui nous a ammené naturellement à regarder l'injection de dépendances. L'approche m'a tout de suite intéressé puisqu'elle permettait de résoudre de manière très intéressante la gestion des relations entre les composants applicatifs. Fort de cette première impression plus que favorable, j'ai regardé en détail la seconde brique de base de Spring, la programmation orientée aspect, puis abordé les différents supports Java / J2EE du framework. Spring m'a permis de mieux (et complètement différemment) penser et structurer l'architecture de mes applications tout en laissant les préoccupations techniques au framework. Enfin un framework permettant de simplifier les développements Java / J2EE!!

Julien Dubois :
Cela fait des années que je m'intéresse à ce type de framework. En 2000 je bossais avec Cocoon 1 (http://cocoon.apache.org), et à l'époque j'avais beaucoup étudié Avalon, le futur noyau de Cocoon 2. Celui-ci a d'ailleurs été remplacé par Spring dans la dernière version de Cocoon (v2.2).
C'est donc tout naturellement que je me suis intéressé à Spring en 2003, lorsque le projet a commencé à être populaire. A l'époque je m'étais embarqué sur les technologies JBoss (EJB 2) et Struts : Spring a été un véritable libérateur pour moi, un retour à un développement propre, pragmatique et intelligent. C'est également ce que je voulais faire partager avec "Spring par la pratique".

Jean-Philippe Retaillé :
Je suis un cas particulier car je ne suis pas un utilisateur de Spring dans le cadre professionnel. La société pour laquelle je travaille impose des contraintes fortes dans l'utilisation de projets Open Source et à l'époque où nous avons dû choisir un framework de type Spring, ce dernier ne les respectaient pas toutes. Heureusement, ces manques ne sont plus d'actualité.

Quelle est la fonctionnalité de Spring dont vous ne pourriez plus vous passer?

Thierry Templier :
La fonctionnalité que je préfère est le support d'accès aux données (intégration de framework de mapping objet relationnel et transactions). En effet, ce dernier offre la possibilité de structurer les composants de ce type de manière optimale tout en mettant en oeuvre des bonnes pratiques. Le développeur n'a plus à se soucier de la gestion des ressources relatives à la source de données. Le peu de lignes de code a mettre en oeuvre est vraiment impressionnant.
J'affectionne également particulièrement le support d'AspectJ dans Spring afin de mettre en oeuvre la programmation orientée aspect avec l'espace de nommage aop.

Julien Dubois :
Ce que je trouve le plus impressionnant actuellement est le support des transactions. C'est incroyable à quel point il est facile d'avoir une couche de service transactionnelle avec Spring, et ce quelque soit votre système d'accès aux données.
Spring permet ainsi d'avoir de manière transparente des transactions sur un ensemble de DAOs, lesquels peuvent être codés en JDBC pur, en Hibernate, en JPA... En utilisant un gestionnaire de transactions distribuées, vous pouvez également avoir des transactions XA entre vos DAOs et des queues JMS, par exemple. Tout cela n'est qu'une question de configuration, au niveau du code vous n'avez qu'à mettre une annotation pour demander à ce que votre service soit transactionnel.
Je recommande en particulier l'utilisation de Spring 2.0 avec Hibernate pour l'accès JDBC, ActiveMQ pour les messages JMS et Atomikos pour la gestion des transactions XA. C'est un ensemble robuste et performant.
Il faut également noter que Spring 2.5 est capable de découvrir automatiquement le gestionnaire de transactions utilisé sous Websphere et Weblogic, ce qui apporte un certain nombre d'améliorations sur ces systèmes.

Jean-Philippe Retaillé :
Pour ma part, je suis assez bluffé par l'intégration des concepts de la Programmation Orientée Aspect dans Spring. En effet, la création d'aspects est loin d'être naturelle pour les développeurs. Avec Spring, ils font de la POA sans le savoir, notamment avec le support des transactions qu'évoque Julien.

Que pensez vous des nouveautés qu'apporte Spring 2.5 ? En particulier, que pensez vous de l'utilisation des annotations pour la gestion des dépendances ?

Thierry Templier :
Spring 2.5 offre d'intéressantes nouvelles fonctionnalités au niveau de la configuration, notamment avec des annotations. Différents espaces de nommage sont également apparus tels que celui relatif à JMS. D'un autre côté, il me semble important de noter que Spring a pris le virage d'OSGi car, désormais, tous les jar de la distribution correspondent à des bundles OSGi. Spring peut donc désormais être complètement utilisé dans des conteneurs de ce type. En parallèle, le projet Spring Dynamic Modules (ex Spring OSGi) continue son chemin afin de simplifier le développement de bundles et apparaît véritablement comme très prometteur.

Julien Dubois :
Spring 2.5 apporte quelques améliorations en termes de performance, ainsi que quelques nouveautés au niveau de la configuration. En particulier je suis très content du nouveau namespace JMS, qui permet une configuration facilitée.

Concernant les annotations, il s'agit juste d'un nouveau moyen de configurer son application : à mon avis cela n'est pas d'un grand intérêt, si ce n'est pour contrer les arguments de ceux qui trouvent que Spring utilise trop de XML.
Je pense qu'il est important de préciser que Spring n'est pas uniquement configurable via XML : le fichier de configuration XML est juste la manière la plus fréquemment utilisée, et c'est une méthode qui a fait ses preuves. Ce fichier XML est également très bien supporté par les IDEs, il y a des plug in sous Eclipse et IDEA pour vous aider à l'écrire ou le refactorer facilement. Mais rien ne vous force à l'utiliser, si vous préférez utiliser des annotations Spring vous propose désormais également cette option.

Jean-Philippe Retaillé :
Le virage OSGi de Spring qu'évoque Thierry est vraiment fondamental à mon sens avec l'émergence de la SOA (Service Oriented Architecture) dans les architectures d'entreprise. En effet, avec des demandes métiers de plus en plus nombreuses, dans des délais de plus en plus courts, impliquant une flexibilité optimale des systèmes d'information, nous avons besoin de construire des applications composites à base de services réutilisables selon des principes qui ne sont pas sans rappeler la révolution du plug&play que le hardware a rencontré il y a plus d'une dizaine d'année. J2EE n'offre pas en standard une souplesse suffisante dans ce domaine et les conteneurs OSGi semblent bien partis pour participer à relever le défi de ce genre d'architectures.

Selon vous, que manque-t-il à Spring ? Quelles seraient les améliorations intéressantes à apporter à Spring ?

Thierry Templier :
Personnellement, je trouve qu'il manque un support plus avancé dans les IDE (Eclipse notamment) afin d'améliorer la productivité au niveau du développement, bien que la dernière version de Spring IDE comble un peu cet aspect notamment au niveau de l'AOP avec un support "à la AJDT". D'après ce que j'ai entendu dire, c'est en marche au niveau de SpringSource...

Julien Dubois :
Je sais que ce n'est pas du tout dans la roadmap de Spring, mais un bon gestionnaire de transactions préconfiguré serait le bienvenu pour tous les gens qui travaillent sous Tomcat ou JBoss.

Jean-Philippe Retaillé :
Je ne puis qu'abonder dans le sens de Thierry. Nous rencontrons de véritables problèmes de productivité sur les projets par rapport à ce qu'on a pu connaître avec les L4G par exemple. Nous avons en quelque sorte sacrifié une partie de notre productivité sur l'autel de l'indépendance (toute relative) vis-à-vis des éditeurs logiciels. Tous les développeurs ne sont pas de experts techniques et peu de choses sont faites en terme d'outillage pour leur simplifier la vie. Les apports de frameworks comme Spring ou Hibernate sont indéniables, tout comme l'apparition des annotations et l'ouverture de Java aux langages dynamiques, mais il faut aller plus loin en outillant mieux le développeur "métier". Toute la difficulté de ce genre d'exercice est de ne pas (re)tomber dans les erreurs du passé que nous avons connus avec les L4G à savoir le "vendor lock-in" et la "leaking abstraction" .

A coté de Spring, quelles sont les API que vous utilisez régulièrement dans vos projets ? Et pourquoi ?

Thierry Templier :
J'utilise couramment des framewoks tels qu'Hibernate pour la persistence des données et DWR afin de mettre en oeuvre Ajax en environnement Java EE.
Au niveau sécurité, j'utilise Acegi qui a l'avantage d'être très flexible et peut être mis en oeuvre sans trop d'impact sur les applications existantes.
Je commence également à utiliser des technologies prometteuses telles que GWT (interfaces Web riches), JPA (persistance des EJB 3) et OSGi dans des prototypes et des projets de veille.

Julien Dubois :
Hibernate, pour la persistance, comme 95% des gens... Dans le "portefeuille Spring" il y a Spring Security (anciennement Acegi Security), qui est utilisé sur presque tous mes projets au boulot, et que je recommande toujours très fréquemment autour de moi. Comme gestionnaire de cache je suis un grand fan d'ehcache : c'est une solution assez simple, mais elle fait bien son boulot. Elle est facile à mettre en œuvre, performante, et ne m'a pour l'instant jamais causé le moindre souci. Pour tout ce qui est AJAX j'aime bien DWR et GWT (encore que ce dernier soit plus qu'une API...). DWR, en particulier, permet de publier automatiquement un bean Spring en JavaScript : vous pouvez ainsi appeler ses méthodes directement depuis le JavaScript de votre page Web. Si vous voulez plus d'informations sur ce sujet, le chapitre de "Spring par la pratique" qui traite de DWR et Spring est téléchargeable gratuitement sur le site d'Eyrolles : Lien2

Jean-Philippe Retaillé :
Essentiellement Struts et Hibernate car ce sont des frameworks éprouvés (voire vieillisant pour le premier) et bien maîtrisés par de nombreux développeurs. Beaucoup d'applications dans le domaine de l'assurance ont des durées de vie très longues et par sécurité nous misons sur des "valeurs sûres" (malheureusement parfois au détriment de solutions plus innovantes)...

Pensez vous que l'intérêt pour Spring risque de baisser avec l'intégration d'un système d'injection de dépendance directement dans JEE ?

Thierry Templier :
Non, je ne pense pas car Spring a un périmètre beaucoup plus large et offre une véritable plateforme afin de développer des applications Java EE plus facilement et de manière très flexible. L'injection de dépendance ainsi que le support de la programmation orientée ne sont que les fondations de cette plateforme.

Julien Dubois :
Il n'y a pas vraiment de rapport entre le système d'IoC relativement simple proposé dans JEE et le portfeuille complet de solutions que propose Spring.
D'autre part, il n'y a pas de concurrence à ce sujet : par exemple les gens de SpringSource ont été depuis le début très impliqués dans l'utilisation de JPA, en particulier via leur travail avec BEA et le projet OpenJPA.

Jean-Philippe Retaillé :
Non car l'intérêt pour Spring ne se limite pas à l'injection de dépendance. Je pense que SpringSource saura bien se positionner par rapport à ce nouveau standard quand il sortira. Par ailleurs, quand un framework rencontre un tel succés, il est très difficile de le remplacer, même par un standard : il suffit de regarder JSF par rapport à Struts.

Retrouvez la suite de l'interview en ligne : Lien2

Tests et benchmark en PHP 5

1. Introduction

1.1. Remerciements

Je souhaiterais, avant toute chose, remercier tous les membres de l'équipe PHP pour l'aide qu'ils m'ont fournie dans la rédaction de cet article.
Je remercie en particulier Yogui et Iubito sans qui je n'aurais pas pu faire cet article.

1.2. Préambule

Cet article est une suite de tests que j'ai réalisée sur ma propre machine et qui permet de se faire une idée sur le temps d'exécution de telle ou telle fonction PHP.

Je tiens à préciser que cet article ne vous fera pas gagner un temps considérable si d'amblée vos pages PHP sont mal codées. Ces mini-optimisations ne remplacent évidemment pas un algorithme bien pensé.

Pensez donc à améliorer les algorithmes de vos pages PHP avant même de penser à "customiser" votre code.

1.3. Conditions de test

Les tests suivants ont été réalisés sur un serveur Apache 2.2.4 (Win32) avec la version 5.2.3 de PHP.

Pour chaque test, nous avons éxecuté 5 fois le même code, à savoir une boucle de 3 millions d'itérations, ceci afin d'avoir des résultats concluants.
La boucle est la suite :

```
<?php
$nb_occur=3000000;

$time_start = microtime(true);

for ($i=0 ; $i<$nb_occur; $i++)
{
        //le test
}

$time_end = microtime(true);
$time = $time_end - $time_start;

echo 'Durée : '.$time.' secondes<br/>';
?>
```

Lorsque les conditions de tests ont été différentes pour des raisons quelconques (par exemple effectuer 3 millions de fois un echo d'un millier de caractères c'est pas très interessant :)), je l'indique dans cet article.

2. 'Apostrophe' VS "Guillemets"

Ce chapitre est inspiré de l'article de Pierre-Baptiste Naigeon Apostrophes ou guillemets : lesquels choisir ? (Lien4).
Je vous conseille donc d'y jeter un coup d'oeil si vous souhaitez avoir de plus amples informations sur la question.

2.1. L'affectation simple

Nous allons tout d'abord commencer par faire une simple affectation de chaîne de caractères à une variable pour voir la différence entre les 2 temps d'exécution.

```
//1° cas
$chaine='Ceci est une chaîne.';

//2° cas
$chaine="Ceci est une chaîne.";
```

	Durée en s
1° cas	1.50106406212
2° cas	1.49910378456

Aucune différence notoire, ... passons au test suivant.

2.2. L'affectation avec variable

Cette fois nous affectons, en plus de la chaine, la variable $i qui correspond au compteur des 3 millions d'itérations.

```
//1° cas
$chaine='Ceci est une variable : '.$i;

//2° cas
$chaine="Ceci est une variable : ".$i;

//3° cas
$chaine="Ceci est une variable : $i";
```

	Durée en s
1° cas	3.96654582024
2° cas	3.97251200676
3° cas	4.95615792274

Voilà une différence et pas des moindres : le troisième cas met 25% plus de temps que le premier et le second qui, eux, sont équivalents. Pourquoi ?
La raison est simple, le serveur PHP décompose la chaîne de caractères dans le troisième cas afin de trouver et interpréter la variable $i, chose qu'il ne fait pas dans les 2 autres cas.
En ce qui concerne l'efficacité pure et dure, la concaténation est donc préférable à l'utilisation de variable à l'intérieur même d'une chaîne de caractères : le premier et le second cas serait à ce moment là ex aequo. Pas tout à fait ...

En effet, il est conseillé d'utiliser les apostrophes car elles permettent d'avoir un code largement plus lisible quand on combine PHP et HTML :

```
 //1° cas
echo '<img src="chemin/vers/fichier/image.jpg"
alt="description image" />';

//2° cas
echo "<img src=\"chemin/vers/fichier/image.jpg\"
alt=\"description image\" />";
```

Dans le premier cas, on utilise les apostrophes pour délimiter la chaîne de caractères ce qui permet d'éviter d'échapper les guillemets dans le code HTML (ce que nous sommes obligés de faire dans le second cas).

Pour conclure, la meilleure solution est donc d'utiliser les apostrophes et de concaténer les variables : cette méthode est non seulement plus rapide mais également plus lisible.

3. L'affichage à l'écran

Cette question de l'affichage de données à l'écran a déjà été traitée en profondeur par Frédéric Bouchery dans cet article : "echo" : lapin ou tortue ? (Lien5)

3.1. Affichage d'une chaine avec une variable

Nous avons d'abord fait le test entre différentes "versions" du echo (voir la documentation sur les chaines de caractères (Lien6)) et le print afin de voir laquelle était la plus performante.

```
//1° cas
echo '. '.$i;

//2° cas
echo '. ',$i;

//3° cas
echo <<<EOT
. $i
EOT;

//4°cas
print '. '.$i;
```

	Durée en s
1° cas	18.6121080875
2° cas	18.3430218697
3° cas	19.0871639729
4° cas	18.7015310287

Bien que les temps d'exécution soient très proches les uns des autres, nous pouvons voir que la version multi-paramètres du echo (2° cas) semble être la plus performante tandis que la syntaxe HereDoc (3° cas) a, elle, plus de mal.

Pour ce qui est du print, il est théoriquement moins rapide qu'un echo car il retourne toujours 1 (cf la doc. officielle de print et plus particulièrement cette brève discussion. (Lien7)) mais nous voyons dans ce test que la différence reste infime.

3.2. "Plusieurs petits echo" VS "un seul gros echo"

J'ai voulu également répondre à une question revenant souvent sur les forums en tout genre "est-il plus rapide de faire un seul gros echo ou plusieurs petits ?".
A priori j'aurais été le premier à répondre "un seul gros" car ce qui fait perdre du temps c'est l'envoi de flux au serveur et il vaut donc mieux envoyer un seul gros paquet que plein de petits ... c'est pas si évident !!!

```
//1°cas : echo constitué de 10 caractères + 1 espace
echo 'llllllllll ';

//2° cas : echo constitué de 100 fois (10 caractères +
1 espace)
echo 'llllllllll llllllllll llllllllll llllllllll
llllllllll llllllllll llllllllll llllllllll
      llllllllll llllllllll llllllllll llllllllll
llllllllll llllllllll llllllllll llllllllll
      llllllllll llllllllll llllllllll llllllllll
llllllllll llllllllll llllllllll llllllllll
      llllllllll llllllllll llllllllll llllllllll
llllllllll llllllllll llllllllll llllllllll
      llllllllll llllllllll llllllllll llllllllll
llllllllll llllllllll llllllllll llllllllll
      llllllllll llllllllll llllllllll llllllllll
llllllllll llllllllll llllllllll llllllllll
      llllllllll llllllllll llllllllll llllllllll
llllllllll llllllllll llllllllll llllllllll
      llllllllll llllllllll llllllllll llllllllll
llllllllll llllllllll llllllllll llllllllll
      llllllllll llllllllll llllllllll llllllllll
llllllllll llllllllll llllllllll llllllllll
      llllllllll llllllllll llllllllll llllllllll
llllllllll llllllllll llllllllll llllllllll
      llllllllll llllllllll llllllllll llllllllll
llllllllll llllllllll llllllllll llllllllll
      llllllllll llllllllll llllllllll llllllllll
llllllllll llllllllll llllllllll llllllllll
      llllllllll llllllllll llllllllll llllllllll ';
```

Pour ce test, j'ai modifié les boucles car l'affichage 3 millions de fois de 1000 caractères est vraiment trop long.
J'ai donc réalisé 2 séries : la première affichant 30 millions de caractères et la seconde affichant 1 million caractères.

	30 millions de caractères (durée en s)	1 million de caractères (durée en s)
1° cas	11.5267999172	0.39734005928
2° cas	10.7492429256	0.329221200943

La théorie est respectée : le second cas est plus rapide bien que la différence ne soit pas si importante.
Par contre, outre le fait que ce soit légèrement plus rapide d'afficher tout dans un gros bloc, c'est surtout bien plus lisible que de faire un echo par ligne comme on le voit souvent :)

Une solution alternative et très efficace est d'utiliser ob_start couplé à ob_end_flush. Il faut appeler ob_start au début de vos pages PHP et ob_end_flush à la fin.
Par exemple sur notre première série (affichage de 30 millions de caractères), la durée d'exécution est de 4.90419578552 secondes soit un gain de temps de plus de 50% !!!
Ceci s'explique car ob_start permet d'envoyer toutes les données, qui devraient être affichées à l'écran, dans le tampon de sortie et c'est seulement lors de l'appel de ob_end_flush que les données sont restituées au navigateur.
Cela permet d'éviter les allers-retours incessants entre le serveur et le navigateur dus à la répétition des echos d'où un gain de temps plus que conséquent.

4. Les variables

Voyons maintenant différents tests sur les variables. Tout d'abord, 2 façons de tester si une variable est nulle puis 2 façons de faire une affectation identique à 2 variables.

4.1. Le test de nullité

Souvent les gens utilisent la fonction is_null afin de savoir si une variable est NULL ou pas. Est-ce vraiment la meilleure solution ?

```
//1° cas
$i===NULL;

//2° cas
is_null($i);
```

	Durée en s
1° cas	1.1594440937
2° cas	2.51942610741

Comme vous pouvez le voir la fonction is_null est 2 fois plus lente que l'opérateur binaire ===.
L'utilisation d'une fonction est généralement plus gourmande en mémoire qu'un opérateur : ceci à cause de l'obligation de sauvegarder l'environnement. Donc, en plus de la perte de performances, l'utlisation de la fonction is_null consomme de la mémoire vive.
Attention tout de même à ne pas confondre une variable nulle et une autre inexistante : les 2 tests ci-dessus renverront tous les 2 des notices si vous testez des variables non définies auparavant dans votre code. Pour pallier ce problème, il faut utiliser les fonctions isset ou empty.

4.2. Déclaration et initialisation de variable

Voyons maintenant les différentes manière de déclarer et d'initialiser plusieurs variables avec la même valeur.

```
//1° cas
$var1 = 'une chaine';
$var2 = 'une chaine';

//2° cas
$var1 = 'une chaine';
$var2 = $var1;

//3° cas
$var1 = $var2 = 'une chaine';
```

	Durée en s
1° cas	2.16453216553
2° cas	2.09558109283
3° cas	2.16578411102

Le temps d'exécution est quasi identique dans les 3 cas. Le second et le troisième cas sont en revanche plus propres au niveau du code (en effet, si la chaîne change il suffit de la modifier à un seul endroit).

Il est à noter que le fait de transtyper une variable augmente légèrement le temps d'exécution.

5. Les structures de contrôle

5.1. Les tests de condition

Nous avons testé ici les 3 différentes manières de traiter les conditions : la structure if/elseif/else, le switch et enfin l'opérateur ternaire.
Nous avons réalisé 2 jeux de tests : un jeu avec seulement 2 possibilités et un autre avec 8 possibilités. Le test consiste simplement à faire un modulo sur le compteur.

```
//1°cas
if($i%2 == 1) $test=0;
else $test=1;

//2°cas
switch($i%2)
{
case 1:
        $test=0;
        break;
default:
        $test=1;
}

//3° cas
$test = (($i%2 == 1) ? 0 : 1);
```

	Durée en s
1° cas	0.887945795059
2° cas	1.14216017723
3° cas	1.089978790283

À la vue du premier jeu de tests, on pourrait dire que l'opérateur classique if/else est le plus rapide mais voyons voir si on rajoute des possibilités.

```
//1°cas
$res=$i%8;
if($res == 0) $test=0;
elseif($res == 1) $test=1;
elseif($res == 2) $test=2;
elseif($res == 3) $test=3;
elseif($res == 4) $test=4;
elseif($res == 5) $test=5;
elseif($res == 6) $test=6;
else $test=7;

//2°cas
$res=$i%8;
switch($res)
{
   case 0:
      $test=0; break;
   case 1:
      $test=1; break;
   case 2:
      $test=2; break;
   case 3:
      $test=3; break;
   case 4:
      $test=4; break;
   case 5:
      $test=5; break;
   case 6:
      $test=6; break;
   default :
```

```
	$test=7;
}

//3° cas
$res=$i%8;
$test = ($res == 0) ? 0 : (
	($res == 1) ? 1 : (
	($res == 2) ? 2 : (
	($res == 3) ? 3 : (
	($res == 4) ? 4 : (
	($res == 5) ? 5 : (
	($res == 6) ? 6 : 7))))));
```

	Durée en s
1° cas	1.92930579185
2° cas	1.83621811867
3° cas	2.26550005913

Si nous rajoutons des possibilités, nous pouvons voir que le switch est plus rapide que les 2 autres solutions.
Ceci nous permet de conclure qu'il vaut mieux utiliser l'opérateur classique if/else quand il y a peu de possibilités et le switch quand il y en a plus.
Il est à noter que l'opérateur ternaire n'est le plus rapide dans aucun des 2 jeux de tests et cela tombe bien étant donné que son code est relativement illisible.

5.2. Les boucles

Nous testons dans ce paragraphe, à travers 2 séries de tests, 3 manières de faire des boucles en PHP 5 : for, while et foreach + range.
Nous avons remplacé la boucle que nous utilisons pour tous les tests par les boucles suivantes :

```
//1° série : $cpt=3000000;
//2° série : $cpt=100000;

//1° cas
for($i=0; $i <$cpt; $i++) $test=$i;

//2° cas
$i=0;
while($i<$cpt)
{
	$test=$i;
	$i++;
}

//3° cas
foreach(range(0,$cpt) as $i) $test=$i;
```

	1° série (durée en s)	2° série (durée en s)
1° cas	1.52313993454	0.0494940280914
2° cas	1.34025406837	0.0419161319733
3° cas	Fatal error: Allowed memory size	0.119401931763

Le while est comme prévu la façon la plus rapide de faire une boucle mais il est évident qu'au niveau de la lisibilité du code, le for est mieux.

Pour ce qui est du foreach + range, il est à oublier pour principalement 2 raisons :

- si vous souhaitez faire de grosses boucles, vous allez tomber sur une "fatal error" (j'ai augmenté le memory_limit jusqu'à 128Mo mais rien n'y change).
- c'est long, trop long ... environ 3 fois plus qu'un while

IV-C. Le "count" et les tableaux dans une boucle
Très souvent on voit des bouts de code avec un count inséré dans le for, c'est mal ! Voyez par vous-même dans ce qui suit.

```
//1° cas
$k=0;
$tableau=array('a', 'b', 'c', 'd', 'e');
$n=count($tableau);
for($i=0; $i<$n; $i++) $k++;

//2° cas
$k=0;
$tableau = array('a', 'b', 'c', 'd', 'e');
for($i=0; $i<count($tableau); $i++) $k++;
```

	Durée en s
1° cas	17.314617157
2° cas	27.6772232056

Le résultat est sans appel, il faut toujours calculer la taille du tableau avant de faire une boucle.
Dans le cas contraire, cette taille est recalculée à chaque tour de boucle, ce qui est une perte de temps considérable.

6. Les fichiers

6.1. La lecture d'un fichier

Pour la lecture nous avons réalisé, à chaque fois, le même test qui consiste à ouvrir, lire puis fermer un fichier 3000 fois. Nous avons réalisé ce test sur 3 fichiers : le premier de 1 octet, le second de 1Ko et le troisième de 1Mo.

6.1.1. Dans une chaine

Voici les 2 codes comparés :

```
//1° cas
$fichier=file_get_contents('test.txt');

//2° cas
$fp =  fopen('test.txt', 'r');
$fichier = fread($fp, filesize('test.txt'));
fclose($fp);
```

	Fichier 1 octet (durée en s)	Fichier 1 Ko (durée en s)	Fichier 1 Mo (durée en s)
1° cas	0.314026117325	0.323743124008	10.8227910995
2° cas	0.27210401535	0.285511016846	9.0904991627

Nous pouvons d'ores et déjà tirer 2 conclusions : lire un fichier "a la mano" est plus rapide qu'en utilisant la fonction file_get_contents.
Le temps d'exécution s'envole lorsque les fichiers à lire deviennent gros. En effet entre un fichier d'un seul octet et un 1000 fois plus gros, le temps est quasiment le même. Par contre, entre un fichier d'un Ko et un d'un Mo, le temps d'exécution est à peu près 30 fois plus long.

6.1.2. Dans un tableau

```
//1° cas
$fichier=file_get_contents('test.txt');
$lignes = split('[\n\r]', $fichier);

//2° cas
$lignes=file('test.txt');

//3° cas
$fp =  fopen('test.txt', 'r');
$fichier = fread($fp, filesize('test.txt'));
fclose($fp);
$lignes = split('[\n\r]', $fichier);
```

	Fichier 1 octet (durée en s)	Fichier 1 Ko (durée en s)	Fichier 1 Mo (durée en s)
1° cas	0.385547161102	0.524013004303	124.7746181493
2° cas	0.379977817535	0.385583868027	11.8984890079
3° cas	0.366604846954	0.483832120895	113.7731289865

Le résultat est sans appel : la lecture d'un fichier dans un tableau est largement plus expéditive avec l'utilisation de la fonction file, surtout lorsque le fichier est "gros".
En effet la solution utilisant la fonction file est, environ, 10 fois plus rapide que les 2 autres sur de gros fichiers.

6.2. L'écriture

Dans cette partie, nous avons écrit de différentes façons dans un fichier.
Tout comme pour la lecture, nous avons réalisé 3 séries : écriture de 1 octet, 1 Ko et enfin 1 Mo dans un fichier que l'on crée auparavant.
Nous avons réalisé 3000 fois chacune de ces opérations. La variable $a_ecrire contient la chaîne de caractères à écrire soit un caractère dans la première série, 1024 dans la seconde et 1048576 dans la troisième et dernière.

```
//1° cas
file_put_contents('test.txt', $a_ecrire);

//2° cas
$fp =  fopen('test.txt', 'w');
fwrite($fp, $a_ecrire);
fclose($fp);
```

	Fichier 1 octet (durée en s)	Fichier 1 Ko (durée en s)	Fichier 1 Mo (durée en s)
1° cas	1.641685075759	1.919305665971	22.27384757997
2° cas	1.568504238128	1.911750719071	18.38060402871

Les résultats restent donc très similaires à la lecture dans un fichier : le temps d'exécution est plus long lorsque l'on utilise la fonction file_put_contents et il augmente exponentiellement avec la taille des données à écrire.

7. Les tableaux

Nous allons maintenant voir ce qui semble être le mieux pour parcourir un tableau.
Pour cela nous allons réaliser les test sur 2 tableaux : un de 10000 cases et un de 5. Le tableau suivant utilisé dans ces test est parcouru 1000 fois :

```
//1 ° série : $nb_cases=10000;
//2° série : $nb_cases=5;

$tab = range(1,$nb_cases);
srand((float)microtime()*1000000);
shuffle($tab);
```

Et voici les différentes façons de parcourir le tableau testé :

```
//1° cas
for ($i=0;$i<10000;$i++) $test=$tableau[$i];

//2° cas
foreach($tableau as $cle => $valeur) $test=$valeur;

//3° cas
foreach($tableau as $valeur) $test=$valeur;

//4° cas
while(list($cle, $valeur) = each($tableau))
$test=$valeur;

//5° cas
$i=0;
while ($i<1000000) $test=$tableau[$i++];
```

	Tableau de 10000 cases (durée en s)	Tableau de 5 cases (durée en ms)
1° cas	6.04399681091	2.54988670349
2° cas	4.66101288795	2.15983390808
3° cas	3.88768601418	0.483832120895
4° cas	0.0336660919189	1.09601020813
5° cas	5.73187505722	2.24590301514

Dans tous les cas, on s'aperçoit que l'utilisation du while couplé aux fonctions list et each est nettement plus rapide que les autres, tout particulièrement sur des grands tableaux.

Attention à l'utilisation du foreach : il travaille sur une copie du tableau spécifié, et pas sur le tableau lui-même. Par conséquent, le pointeur de tableau n'est pas modifié, comme il le serait avec la fonction each(), donc les modifications faites dans le tableau ne seront pas prises en compte dans le tableau original.

8. Les remplacements dans les chaînes de caractères

Il existe de multiples manières pour remplacer des caractères dans une chaîne en PHP mais laquelle est la plus rapide ? Tout d'abord, si nous regardons la documentation PHP officielle de ereg_replace, celle-ci dit : "preg_replace(), qui utilise la syntaxe des expressions rationnelles compatibles PERL, est une alternative plus rapide de ereg_replace()."
Donc sans faire de tests, nous savons déjà que preg_replace va plus vite que ereg_replace.
Ensuite, comme l'indique encore une fois la documentation, preg_replace utilise des expressions régulières tandis que str_replace prend une chaîne de caractères simple en paramètre. La fonction str_replace est donc plus rapide que preg_replace.

Voyons donc str_replace en oeuvre.
Nous avons réalisé 2 séries de tests : la première consiste en remplacer "abcxyz" par "bbcyyz" tandis que la seconde remplace "Hello ! abcdefghijklmnopqrstuvwxyz" par "BOnjOUr ! AbCdEfGhIjKlMnOpQrStUvWxYz".
Pour la seconde série nous n'avons réalisé qu'un million d'itérations au lieu des 3 millions habituels car les temps étaient

trop élevés.
Voici les 4 utilisations de str_replace testées :

```
1° série
//$a = 'abcxyz';

//1° cas
$a = str_replace('x','y',$a);
$a = str_replace('a','b',$a);

//2° cas
$a = str_replace('x','y', str_replace('a','b',$a));

//3° cas
$a = str_replace(array('x','a'), array('y','b'), $a);

//4° cas
$a = strtr($a, 'ax', 'by');

2° série
//$a = 'Hello ! abcdefghijklmnopqrstuvwxyz'

//1° cas
$a = str_replace('Hello','Bonjour',$a);
$a = str_replace('a','A',$a);
$a = str_replace('c','C',$a);
$a = str_replace('e','E',$a);
$a = str_replace('g','G',$a);
$a = str_replace('i','I',$a);
$a = str_replace('k','K',$a);
$a = str_replace('m','M',$a);
$a = str_replace('o','O',$a);
$a = str_replace('q','Q',$a);
$a = str_replace('s','S',$a);
$a = str_replace('u','U',$a);
$a = str_replace('w','W',$a);
$a = str_replace('y','Y',$a);

//2° cas
$a = str_replace('a','A', str_replace('c','C',
   str_replace('e','E',
   str_replace('g','G',
   str_replace('i','I',
   str_replace('k','K',
   str_replace('m','M',
   str_replace('o','O',
   str_replace('q','Q',
   str_replace('s','S',
   str_replace('u','U',
   str_replace('w','W',
   str_replace('y','Y',
   str_replace ('Hello','Bonjour', $a)))))))))))));

//3° cas
$a = str_replace(
   array('Hello','a','c','e','g','i','k','m','o','q','s
','u','w','y'),
   array('Bonjour','A','C','E','G','I','K','M','O','Q',
'S','U','W','Y'),
   $a);

//4° cas
$a = strtr(str_replace('Hello', 'Bonjour', $a),
'acegikmoqsuwy', 'ACEGIKMOQSUWY');
```

	1° série (durée en s)	2° série (durée en s)
1° cas	8.0842359066	57.0145347117
2° cas	7.71057006836	17.7155649662
3° cas	10.6429069042	15.9177130699
4° cas	4.14906597137	3.26361989975

Aucun doute à la vue de ces résultats : la fonction strtr est bien plus rapide que la fonction str_replace tout particulièrement lorsqu'il s'agit de modifier de nombreux caractères.
Un petit bémol cependant : strtr permet de modifier des caractères par d'autres caractères mais il ne peut pas remplacer de "mots" d'où l'utilisation couplé de strtr et de str_replace dans la seconde série.

Juste pour avoir un ordre d'idée, voyons quels résultats nous aurions obtenus avec preg_replace. Prenons le code suivant :

```
$patterns[0] = '/a/';
$patterns[1] = '/x/';

$replacements[0] = 'b';
$replacements[1] = 'y';
$a = preg_replace($patterns, $replacements, $a);
```

Ce code affiche le temps d'exécution suivant : Durée : 19.8709621429 secondes ... sans commentaires.

Nous pouvons donc conclure que le meilleur moyen de remplacer des caractères dans une chaîne de caractères est l'utilisation de strtr couplé à str_replace s'il y a nécessité à remplacer des "mots" entiers.

9. Conclusion

Comme vous avez pu le voir tout au long de l'article, il y a des techniques de programmation afin d'optimiser son code et de le rendre plus performant.
Toutes ces astuces de programmation ne pallieront jamais le manque de conception dans vos programmes ou bien tout simplement des algorithmes bâclés.
Pensez également qu'en optimisant le code à outrance pour gagner quelques microsecondes, il risque de devenir totalement illisible ce qui peut entraîner de sérieux problèmes de maintenance dans le cadre d'un travail en équipe.
Il est évident que ces tests ne sont en rien des preuves et permettent seulement de se faire une idée sur certaines questions récurrentes.

Retrouvez le tutoriel de Matthieu Fernandez en ligne : Lien8

Personnalisation d'un formulaire XHTML avec CSS

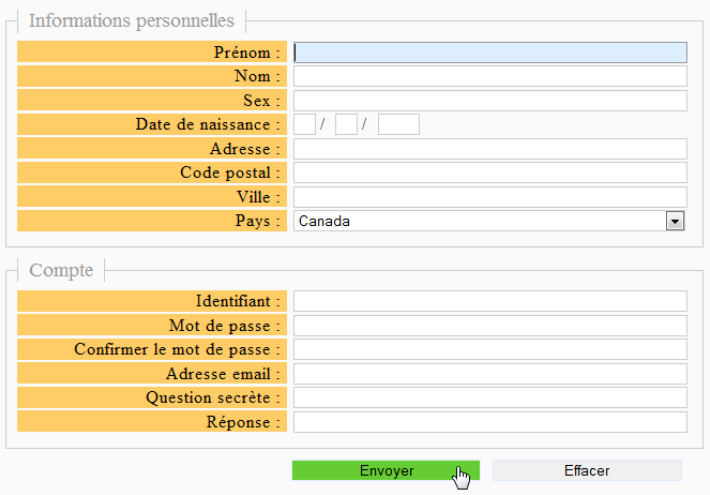
Cet article vous explique comment mettre en forme un formulaire avec du CSS sans oublier les balises label, fieldset et optgroup.

1. Introduction

Je vais vous présenter étape par étape le même formulaire mais avec un petit quelque chose en plus. Au final nous aurons un beau formulaire CSS.

Il y aura d'abord une partie XHTML pour rappeler certaines balises souvent oubliées.
Ensuite viendra la mise en forme du formulaire grâce à du CSS.

Au final, dans un navigateur supportant toutes les propriétés CSS utilisées dans l'article, ça donnera un formulaire comme ceci :



2. Partie XHTML

2.1. Formulaire de base

Le formulaire de base que je vais utiliser est tout ce qu'il y a de plus simple à réaliser.

```
<form action="#" method="post">
    <p>Prénom : <input type="text" name="firstname" /></p>
    <p>Nom : <input type="text" name="lastname" /></p>
    <p>Sexe : <input type="text" name="gender" /></p>
    <p>Date de naissance : <input type="text" name="day" /><input type="text" name="month" /><input type="text" name="year" /></p>
    <p>Adresse : <input type="text" name="adress" /></p>
    <p>Code postal : <input type="text" name="postalCode" /></p>
    <p>Ville : <input type="text" name="city" /></p>
    <p>Pays :
        <select id="form_country" name="country">
            <option value="ca">Canada</option>
            <option value="us">États-Unis</option>
            <option value="be">Belgique</option>
            <option value="fr">France</option>
        </select>
    </p>
    <p>Identifiant : <input type="text" name="login" /></p>
    <p>Mot de passe : <input type="password" name="password" /></p>
    <p>Confirmer le mot de passe : <input type="password" name="password2" /></p>
    <p>Adresse email : <input type="text" name="mail" /></p>
    <p>Question secrète : <input type="text" name="question" /></p>
    <p>Réponse : <input type="text" name="response" /></p>
    <p><input type="submit" name="submit" /></p>
</form>
```

2.2. La balise label

La balise label permet d'associer un champ du formulaire à sa description. Elle permet aussi de donner le focus à ce champ quand on clique sur son contenu.
Il existe deux manières correctes d'utiliser cette balise. Soit vous entourez le champ et sa description, soit vous entourez seulement la description. C'est cette dernière solution qui sera utilisée.

```
<form action="#" method="post">
    <p>
        <label for="form_firstname">Prénom : </label>
        <input type="text" id="form_firstname" name="firstname" />
    </p>
    <p>
        <label for="form_lastname">Nom : </label>
        <input type="text" id="form_lastname" name="lastname" />
    </p>
    <p>
        <label for="form_gender">Sexe : </label>
        <input type="text" id="form_gender" name="gender" />
    </p>
    <p>
        <label for="form_birthday">Date de naissance : </label>
        <input type="text" id="form_birthday" name="day" /><input type="text" name="month" /><input type="text" name="year" />
    </p>
    <p>
        <label for="form_address">Adresse : </label>
        <input type="text" id="form_address" name="address" />
    </p>
    <p>
        <label for="form_postal_code">Code postal : </label>
        <input type="text" id="form_postal_code" name="postal_code" />
    </p>
```

```
    <p>
        <label for="form_city">Ville : </label>
        <input type="text" id="form_city" name="city"
/>
    </p>
    <p>
        <label for="form_country">Pays : </label>
        <select id="form_country" name="country">
            <option value="ca">Canada</option>
            <option value="us">États-Unis</option>
            <option value="be">Belgique</option>
            <option value="fr">France</option>
        </select>
    </p>
    <p>
        <label for="form_login">Identifiant : </label>
        <input type="text" id="form_login" name="login"
/>
    </p>
    <p>
        <label for="form_password">Mot de passe :
</label>
        <input type="password" id="form_password"
name="password" />
    </p>
    <p>
        <label for="form_password2">Confirmer le mot de
passe : </label>
        <input type="password" id="form_password2"
name="password2" />
    </p>
    <p>
        <label for="form_mail">Adresse email : </label>
        <input type="text" id="form_mail" name="mail"
/>
    </p>
    <p>
        <label for="form_question">Question secrète :
</label>
        <input type="text" id="form_question"
name="question" />
    </p>
    <p>
        <label for="form_response">Réponse : </label>
        <input type="text" id="form_response"
name="response" />
    </p>
    <p>
        <input type="submit" name="submit" />
    </p>
</form>
```

2.3. La balise fieldset

La balise fieldset permet d'entourer un groupe logique de champs. Cette balise est complémentée par la balise legend qui permet de donner un nom à ce groupe logique.

```
<form action="#" method="post">
    <fieldset>
        <legend>Informations personnelles</legend>
        <p>
            <label for="form_firstname">Prénom :
</label>
            <input type="text" id="form_firstname"
name="firstname" />
        </p>
        <p>
            <label for="form_lastname">Nom : </label>
            <input type="text" id="form_lastname"
name="lastname" />
        </p>
        <p>
            <label for="form_gender">Sexe : </label>
            <input type="text" id="form_gender"
name="gender" />
        </p>
        <p>
            <label for="form_birthday">Date de
naissance : </label>
            <input type="text" id="form_birthday"
name="day" /><input type="text" name="month" /><input
type="text" name="year" />
        </p>
        <p>
            <label for="form_address">Adresse :
</label>
            <input type="text" id="form_address"
name="address" />
        </p>
        <p>
            <label for="form_postal_code">Code postal :
</label>
            <input type="text" id="form_postal_code"
name="postal_code" />
        </p>
        <p>
            <label for="form_city">Ville : </label>
            <input type="text" id="form_city"
name="city" />
        </p>
        <p>
            <label for="form_country">Pays : </label>
            <select id="form_country" name="country">
                <option value="ca">Canada</option>
                <option value="us">États-Unis</option>
                <option value="be">Belgique</option>
                <option value="fr">France</option>
            </select>
        </p>
    </fieldset>

    <fieldset>
        <legend>Compte</legend>
        <p>
            <label for="form_login">Identifiant :
</label>
            <input type="text" id="form_login"
name="login" />
        </p>
        <p>
            <label for="form_password">Mot de passe :
</label>
            <input type="text" id="form_password"
name="password" />
        </p>
        <p>
            <label for="form_password2">Confirmer le
mot de passe : </label>
            <input type="password" id="form_password2"
name="password2" />
        </p>
        <p>
            <label for="form_mail">Adresse email :
</label>
            <input type="password" id="form_mail"
name="mail" />
        </p>
        <p>
            <label for="form_question">Question secrète
: </label>
            <input type="text" id="form_question"
name="question" />
        </p>
        <p>
            <label for="form_response">Réponse :
</label>
```

```
            <input id="form_response" name="response"
/>
        </p>
    </fieldset>

    <p>
        <input type="submit" name="submit" />
    </p>
</form>
```

2.4. La balise optgroup

La balise optgroup s'utilise dans une balise select et permet de grouper les options de cette dernière.
Pour spécifier le nom du groupe, il faut utiliser l'attribut label de optgroup.

```
<form id="monForm" action="#" method="post">
    <fieldset>
        <legend>Informations personnelles</legend>
        <p>
            <label for="form_firstname">Prénom :
</label>
            <input type="text" id="form_firstname"
name="firstname" />
        </p>
        <p>
            <label for="form_lastname">Nom : </label>
            <input type="text" id="form_lastname"
name="lastname" />
        </p>
        <p>
            <label for="form_gender">Sexe : </label>
            <input type="text" id="form_gender"
name="gender" />
        </p>
        <p>
            <label for="form_birthday">Date de
naissance : </label>
            <input type="text" id="form_birthday"
name="day" /><input type="text" name="month" /><input
type="text" name="year" />
        </p>
        <p>
            <label for="form_address">Adresse :
</label>
            <input type="text" id="form_address"
name="address" />
        </p>
        <p>
            <label for="form_postal_code">Code postal :
</label>
            <input type="text" id="form_postal_code"
name="postal_code" />
        </p>
        <p>
            <label for="form_city">Ville : </label>
            <input type="text" id="form_city"
name="city" />
        </p>
        <p>
            <label for="form_country">Pays : </label>
            <select id="form_country" name="country">
                <optgroup label="Amérique">
                    <option value="ca">Canada</option>
                    <option value="us">États-
Unis</option>
                </optgroup>
                <optgroup label="Europe">
                    <option
value="be">Belgique</option>
                    <option value="fr">France</option>
                </optgroup>
            </select>
        </p>
    </fieldset>

    <fieldset>
    <legend>Compte</legend>
        <p>
            <label for="form_login">Identifiant :
</label>
            <input type="text" id="form_login"
name="login" />
        </p>
        <p>
            <label for="form_password">Mot de passe :
</label>
            <input type="password" id="form_password"
name="password" />
        </p>
        <p>
            <label for="form_password2">Confirmer le
mot de passe : </label>
            <input type="password" id="form_password2"
name="password2" />
        </p>
        <p>
            <label for="form_mail">Adresse email :
</label>
            <input type="text" id="form_mail"
name="mail" />
        </p>
        <p>
            <label for="form_question">Question secrète
: </label>
            <input type="text" id="form_question"
name="question" />
        </p>
        <p>
            <label for="form_response">Réponse :
</label>
            <input type="text" id="form_response"
name="response" />
        </p>
    </fieldset>

    <p>
        <input type="submit" name="submit" />
    </p>
</form>
```

Voir le formulaire : Lien9

3. Partie CSS

Attention : Certaines propriétés CSS que j'ai utilisées ne sont pas compatibles IE 6 et ne marchent pas très bien sur IE 7. (exemple: focus qui n'est pas pris en compte et hover qui ne l'est qu'à moitié).

3.1. Fieldset

Ici, on va fixer la taille maximale du formulaire pour ne pas que les cadres des éléments fieldset soient trop grands. On les écarte aussi un peu entre eux pour bien séparer les groupes logiques. Ensuite on peut changer l'aspect de la bordure et la rendre plus discrète. Il y a moyen de mettre en forme l'élément legend, par exemple un texte plus grand, en gras, avec des bordures gauche et droite comme dans l'exemple ci-dessous. On applique un hover à l'élément fieldset pour faire joli.

```
#monForm
{
```

```
	width: 60%;
}

#monForm p
{
	margin: 2px 0;
}

/* fieldset , legend */
#monForm fieldset
{
	margin-bottom: 10px;
	border: #CCC 1px solid;
}

#monForm fieldset:hover
{
	background-color: #FFF;
}

#monForm fieldset legend
{
	padding: 0 10px;
	border-left: #CCC 1px solid;
	border-right: #CCC 1px solid;
	font-size: 1.2em;
	color: #999;
}
```

3.2. Label

Pour bien séparer les éléments label des champs, on va leur mettre une couleur de fond. Il ne faut pas oublier d'annuler cette couleur de fond pour le label correspondant aux boutons. On peut aussi mettre un hover sur les éléments label, toujours pour faire joli.
Si on veut pouvoir spécifier une taille pour les éléments label, il faut mettre l'affichage avec display en block. Pour aligner les champs de texte avec leur label on met en float: left les éléments label.
On aligne ensuite le texte à droite et on applique au texte un letter-spacing pour une meilleure visibilité.

```
/* Label */
#monForm label
{
	background-color: #FFCC66;
	display: block;
	width: 39%;
	float: left;
	padding-right: 1%;
	text-align: right;
	letter-spacing: 1px;
}

#monForm label:hover
{
	font-weight: bold;
}

#monForm .form_label_nostyle
{
	background: none;
}
```

3.3. Input

Naturellement, on va modifier la taille des champs pour les avoir tous à la même longueur. On met en couleur le fond des champs au survol de la souris grâce à un hover et on garde ce fond coloré quand on est dans le champ grâce à un focus.

```
/* Input */
#monForm input, #monForm select
{
	margin-left: 1%;
	width: 58%;
	border: #CCC 1px solid;
}

#monForm input:hover, #monForm select:hover, #monForm
input:focus, #monForm select:focus
{
	border: #999 1px solid;
	background-color: #DDEEFF;
}

#monForm .form_input_day_month
{
	width: 3%;
}

#monForm .form_input_year
{
	width: 6%;
}
```

3.4. Bouton submit et reset

On met en couleur verte et rouge respectivement le bouton submit et le bouton reset. Ensuite il y a moyen de changer le pointeur de forme au survol de la souris sur le bouton grâce à cursor.

```
/* button submit */
#monForm input[type="submit"]
{
	border: #DDEEFF 1px solid;
	width: 27%;
}

#monForm input[type="submit"]:hover
{
	background-color: #66CC33;
	cursor: pointer;
}

#monForm input[type="reset"]
{
	border: #DDEEFF 1px solid;
	width: 27%;
}

#monForm input[type="reset"]:hover
{
	background-color: #E6484D;
	cursor: pointer;
}
```

Voir le formulaire : Lien10

4. Conclusion

Les quelques balises présentées dans cet article ne sont pas obligatoires mais permettent à votre formulaire d'être accessible et d'être mieux présenté.
Quant à la mise en forme et aux quelques effets ajoutés au formulaire, il n'y a guère besoin de Javascript, le CSS suffit de lui-même.

Retrouvez l'article d'Adrien Pellegrini en ligne : Lien11

Personnalisation d'un formulaire XHTML en Javascript

Ce document a pour but de vous expliquer comment personnaliser le moindre élément des formulaires (X)HTML.

1. Introduction

N'aviez-vous jamais rêvé de personnaliser intégralement vos formulaires (X)HTML, du simple champ texte en passant par les boutons radio, les listes déroulantes et même les champs file ?
Cet article va vous montrer une des nombreuses (pas si nombreuses) méthodes de réaliser ce tour de force.

Pour une meilleure compatibilité avec les différents navigateurs et différentes configurations, nous allons essayer de limiter le Javascript au maximum, bien qu'il soit nécessaire pour certains éléments. Aussi, ces derniers ne fonctionneront pas correctement si l'internaute désactive Javascript.
Rassurons-nous, avec l'avènement du Web 2 et ses sites qui tendent vers une utilisation parfois abusive de l'AJAX, peu d'internautes désactivent Javascript sous peine de ne plus pouvoir naviguer sur de nombreux sites.

De plus, je vous conseille fortement d'utiliser un DOCTYPE (Lien12) complet pour vos développements, ce qui gommera les différences d'interprétation des propriétés CSS utilisées dans cet article entre navigateurs.

2. Les inputs

2.1. text, password

Nous allons commencer simple avec la personnalisation des champs de type text et password.

Tout d'abord nous nous contenterons d'habiller les champs :

(X)HTML
```
<fieldset id="conteneurInput">
    <label for="login">Login :</label>
    <input type="text" name="login" id="login" />
    <label for="password">Mot de passe :</label>
    <input type="password" name="password"
id="password" />
</fieldset>
```

CSS
```
label
{
    background      : #fc6;
    padding-right   : 5px;
    margin-right    : 3px;
}

#conteneurInput input
{
    border          : 1px solid #999;
    background      : #def;
}
```

Voir le résultat : Lien13

Ce qui est bien, mais pas top.

Pour donner plus d'allure à nos champs, arrondir les angles serait parfait. Pour cela, il nous faut créer quatre images qui nous serviront de coins.
En outre, une modification conséquente du code s'avère obligatoire :

(X)HTML
```
<fieldset id="conteneurInput">
    <label for="login">Login :</label>
    <div>
        <span class="top-left"> </span>
        <span class="bottom-left"> </span>
        <span class="bottom-right"> </span>
        <span class="top-right"> </span>
        <input type="text" name="login" id="login" />
    </div>
    <label for="password">Mot de passe :</label>
    <div>
        <span class="top-left"> </span>
        <span class="bottom-left"> </span>
        <span class="bottom-right"> </span>
        <span class="top-right"> </span>
        <input type="password" name="password"
id="password" />
    </div>
</fieldset>
```

CSS
```
label
{
    display         : block;
    padding-right   : 5px;
    float           : left;
    background      : #fc6;
    margin-right    : 3px;
}

#conteneurInput div
{
    background      : #def;
    position        : relative;
    border          : 1px solid #999;
    text-align      : center;
    float           : left;
}

#conteneurInput input
{
    background      : none;
    border          : 0;
    padding         : 0 6px;
    width           : 130px;
```

```
}

span.top-left
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    top             : -1px;
    left            : -1px;
    background      : url("top-left.gif");
}

span.bottom-left
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    bottom          : -1px;
    left            : -1px;
    background      : url("bottom-left.gif");
}

span.bottom-right
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    bottom          : -1px;
    right           : -1px;
    background      : url("bottom-right.gif");
}

span.top-right
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    top             : -1px;
    right           : -1px;
    background      : url("top-right.gif");
}
```

Voir le résultat : Lien14

C'est bien mieux !

2.2. radio

La personnalisation des boutons radio (ou radiobuttons), quant à elle, requiert une petite partie de Javascript, celle qui permettra de permuter les images qu'on va utiliser pour présenter nos boutons.

Tout d'abord, le code HTML :

(X)HTML
```
<fieldset id="radio">
    <p>
        <label for="mlle">Mlle :</label>
        <input type="radio" id="mlle" name="civilite2"
onclick="turnImgRadio(this)" />
        <img src="radio1.gif" id="img_radio_mlle"
alt="Bouton radio" />
    </p>
    <p>
        <label for="mme">Mme :</label>
        <input type="radio" id="mme" name="civilite2"
onclick="turnImgRadio(this)" />
        <img src="radio1.gif" id="img_radio_mme"
alt="Bouton radio" />
    </p>
    <p>
        <label for="mr">Mr :</label>
        <input type="radio" id="mr" name="civilite2"
onclick="turnImgRadio(this)" />
        <img src="radio1.gif" id="img_radio_mr"
alt="Bouton radio" />
    </p>
</fieldset>
```

Comme on s'en doute à la lecture de ce code, l'image présente sous le bouton radio sera celle qui le remplacera et l'attribut onclick du bouton radio servira à permuter cette image.

Comment faire apparaître l'image à la place du bouton radio ? En le masquant bien sûr.

CSS
```
label
{
    padding-right   : 3px;
    margin-right    : 3px;
    background      : #fc6;
    vertical-align  : top;
}

#conteneurRadio p
{
    position        : relative;
    float           : left;
    margin          : 0;
}

#conteneurRadio input
{
    opacity         : 0; /* pour !IE */
    filter          : alpha(opacity=0); /* pour IE */
    width           : 20px;
    height          : 20px;
    position        : absolute;
    right           : 0;
    top             : 0;
}
```

Voilà nos boutons radio invisibles, mais actifs !
Autant le dire tout de suite, ce code CSS n'est pas valide W3C. Vous pouvez à la rigueur mettre le filtre Microsoft dans un commentaire conditionnel, mais il faudra attendre le CSS3 pour voir apparaître la propriété opacity.

Il ne reste plus qu'à créer notre fonction Javascript qui va simplement modifier l'image sélectionnée :

Javascript
```
function turnImgRadio(objRadio)
{
    var t_img =
document.getElementById('conteneurRadio').getElementsBy
TagName('img');

    for (var i = 0; i < t_img.length; i++)
    {
        t_img[i].src = 'radio1.gif';
    }

    var img = document.getElementById('img_radio_' +
objRadio.id);
```

```
    img.src = 'radio2.gif';
}
```

Voir le résultat : Lien15

2.3. checkbox

Pas de grands changements avec le paragraphe précédent pour personnaliser les cases à cocher (ou checkboxes), si ce n'est la fonction Javascript qui perd quelques lignes.

(X)HTML

```
<fieldset id="conteneurCheckbox">
    <p>
        <label for="beau">Beau :</label>
        <input type="checkbox" id="beau" name="qualite"
onclick="turnImgCheck(this)" />
        <img src="check1.gif" id="img_check_beau"
alt="Checkbox" />
    </p>
    <p>
        <label for="fort">Fort :</label>
        <input type="checkbox" id="fort" name="qualite"
onclick="turnImgCheck(this)" />
        <img src="check1.gif" id="img_check_fort"
alt="Checkbox" />
    </p>
    <p>
        <label for="intelligent">Intelligent :</label>
        <input type="checkbox" id="intelligent"
name="qualite" onclick="turnImgCheck(this)" />
        <img src="check1.gif"
id="img_check_intelligent" alt="Checkbox" />
    </p>
</fieldset>
```

CSS

```
label
{
    padding-right   : 3px;
    margin-right    : 3px;
    background      : #fc6;
    vertical-align  : top;
}

#conteneurCheckbox p
{
    position        : relative;
    float           : left;
    margin          : 0;
}

#conteneurCheckbox input
{
    opacity         : 0; /* pour !IE */
    filter          : alpha(opacity=0); /* pour IE */
    width           : 20px;
    height          : 20px;
    position        : absolute;
    right           : 0;
    top             : 0;
}
```

Javascript

```
function turnImgCheck(objCheck)
{
    var img = document.getElementById('img_check_' +
objCheck.id);
    var t = img.src.split('/');
    img.src = (t[t.length-1] == 'check2.gif') ?
'check1.gif' : 'check2.gif';
}
```

Voir le résultat : Lien16

2.4. button, submit, reset

Rien de bien compliqué dans la personnalisation des boutons, comme vous pourrez en juger :

(X)HTML

```
<fieldset id="conteneurButton">
    <p>
        <input type="button" name="envoyer"
value="Envoyer" class="ok" />
        <input type="reset" name="effacer"
value="Effacer" class="nok" />
    </p>
</fieldset>
```

CSS

```
#conteneurButton input.ok
{
    border          : 1px solid #def;
    background      : #6c3;
    cursor          : pointer;
    padding         : 3px 30px;
    margin          : 0 10px;
}

#conteneurButton input.nok
{
    border          : 1px solid #def;
    background      : #e6484d;
    cursor          : pointer;
    padding         : 3px 30px;
    margin          : 0 10px;
}
```

Voir le résultat : Lien17

2.5. file

Est-ce vraiment possible de personnaliser un input de type file ? Et même modifier ce maudit texte "Parcourir..." ?

Cette opération est certes plus complexe à mettre en oeuvre que celle du paragraphe précédent, mais elle repose sur la même technique que celle des checkboxes et radiobuttons : masquer l'input et non l'émuler.

(X)HTML

```
<fieldset id="conteneurFile">
    <div id="divFile">
        <input type="text" id="input_text_file"
class="inputText" readonly="readonly" />
        <input type="file" onmousedown="return false"
onkeydown="return false" class="inputFile"
onchange="document.getElementById('input_text_file').va
lue = this.value" />
        <span>Ajouter...</span>
    </div>
</fieldset>
```

CSS

```
#file #divFile
{
    position        : relative;
    width           : 250px;
    text-align      : right;
}
```

```
#conteneurFile .inputFile
{
    opacity         : 0; /* pour !IE */
    filter          : alpha(opacity=0); /* pour IE */
    position        : absolute;
    right           : 0;
    top             : 0;
}

#conteneurFile .inputText
{
    border          : 1px solid #999;
    padding         : 0px 6px;
    background      : #def;
    width           : 130px;
}

#conteneurFile span
{
    border          : 1px solid #def;
    background      : #ffc;
    width           : 80px;
    padding         : 1px 10px;
}
```

Voir le résultat : Lien18

Les évènements onmousedown et onkeydown sur le champ file interdisent la saisie de texte dans ce champ invisible, ça pourrait perturber l'internaute de voir qu'il saisit du texte qui ne s'affiche pas.
L'évènement onchange, quant à lui, fait juste apparaître le nom du fichier dans notre champ texte, pour que l'internaute ne soit pas déboussolé.

Pour positionner correctement le bouton "Parcourir..." invisible par rapport au texte de substitution, vous pouvez modifier l'opacité pour voir où vous en êtes.

À partir de ce code, on peut évidemment imaginer plein de façons différentes de personnaliser notre input file, comme mettre une image à la place du texte "Ajouter...", modifier l'apparence du champ texte avec des angles arrondis selon la méthode du premier paragraphe de cet article, etc.

À vous d'imaginer le meilleur.

3. Le textarea

La méthode de personnalisation d'un textarea est strictement similaire à celle utilisée pour les champs texte.

Voici donc le code pour avoir des angles normaux :

(X)HTML
```
<fieldset id="conteneurTextarea">
    <label for="message">Message :</label>
    <textarea name="message" id="message" cols="30"
rows="5"></textarea>
</fieldset>
```

CSS
```
label
{
    background      : #fc6;
    padding-right   : 5px;
    margin-right    : 3px;
}

#conteneurTextarea textarea
{
    border          : 1px solid #999;
    background      : #def;
    vertical-align  : top;
    overflow        : auto;
}
```

Voir le résultat : Lien19

Et le code pour avoir des angles arrondis, toujours avec nos quatre images :

(X)HTML
```
<fieldset id="conteneurTextarea">
    <label for="message">Message :</label>
    <div>
        <span class="top-left"> </span>
        <span class="bottom-left"> </span>
        <span class="bottom-right"> </span>
        <span class="top-right"> </span>
        <textarea name="message" id="message" cols="30"
rows="5"></textarea>
    </div>
</fieldset>
```

CSS
```
label
{
    display         : block;
    padding-right   : 5px;
    float           : left;
    background      : #fc6;
    margin-right    : 3px;
}

#conteneurTextarea div
{
    background      : #def;
    position        : relative;
    border          : 1px solid #999;
    text-align      : center;
    float           : left;
}

#conteneurTextarea textarea
{
    background      : none;
    border          : 0;
    padding         : 0 6px;
    width           : 300px;
    overflow        : auto;
}

span.top-left
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    top             : -1px;
    left            : -1px;
    background      : url("top-left.gif");
}

span.bottom-left
{
    position        : absolute;
```

```
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    bottom          : -1px;
    left            : -1px;
    background      : url("bottom-left.gif");
}

span.bottom-right
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    bottom          : -1px;
    right           : -1px;
    background      : url("bottom-right.gif");
}

span.top-right
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    top             : -1px;
    right           : -1px;
    background      : url("top-right.gif");
}
```

Voir le résultat : Lien21

4. Le select

Contrairement aux autres éléments qui n'avaient quasiment pas besoin de Javascript, pour pouvoir personnaliser un select, au contraire, il est indispensable.
En effet, pour afficher une liste sur un clic, on est forcé d'utiliser les évènements Javascript.

Dans l'exemple ci-dessous, les angles seront arrondis à la manière décrite dans le premier paragraphe, vous pouvez évidemment vous en dispenser ou même l'améliorer.

(X)HTML

```
<fieldset id="conteneurSelect">
    <label>Pays :</label>
    <div class="inputsSelect">
        <span class="top-left"> </span>
        <span class="bottom-left"> </span>
        <span class="bottom-right"> </span>
        <span class="top-right"> </span>
        <p class="selects"
onclick="showHideSelect('listeSelect1')">-- Choisissez
--</p>
        <ul id="listeSelect1">
            <li><a href="javascript:void(0)"
onclick="validAndHide('', this, 'countryCode',
'select1')">-- Choisissez --</a></li>
            <li><a href="javascript:void(0)"
onclick="validAndHide('FR', this, 'countryCode',
'select1')">France</a></li>
            <li><a href="javascript:void(0)"
onclick="validAndHide('DE', this, 'countryCode',
'select1')">Allemagne</a></li>
            <li><a href="javascript:void(0)"
onclick="validAndHide('IT', this, 'countryCode',
'select1')">Italie</a></li>
            <li><a href="javascript:void(0)"
onclick="validAndHide('VG', this, 'countryCode',
'select1')">Saint-Vincent-et-les Grenadines</a></li>
        </ul>
    </div>
    <input type="hidden" name="countryCode"
id="countryCode" />
</fieldset>
```

Le input de type hidden sert à récupérer la valeur sélectionnée pour pouvoir l'envoyer à la soumission du formulaire.

CSS

```
label
    {
    display         : block;
    padding-right   : 5px;
    float           : left;
    background      : #fc6;
    margin-right    : 3px;
}

p
{
    margin          : 0;
}

#conteneurSelect .inputsSelect
{
    background      : #def url("fleche.gif") right
center no-repeat;
    position        : relative;
    border          : 1px solid #999;
    text-align      : center;
    float           : left;
}

.inputsSelect .selects
{
    padding         : 3px 14px 3px 3px;
    font            : normal 12px verdana;
    cursor          : default;
    width           : 95px;
    white-space     : nowrap;
    overflow        : hidden;
}

.inputsSelect ul
{
    position        : absolute;
    text-align      : left;
    border          : 1px solid #999;
    white-space     : nowrap;
    font            : normal 12px verdana;
    padding         : 5px;
    display         : none;
    background      : #eff7ff;
    z-index         : 100;
    list-style      : none;
    margin          : 0;
}

.inputsSelect ul li a
{
    display         : block;
    cursor          : default;
    color           : #000;
    text-decoration : none;
    background      : #eff7ff;
    width           : 100%;
}

.inputsSelect ul li a:hover
{
```

```
    color           : #fff;
    background      : #093e6d;
}

span.top-left
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    top             : -1px;
    left            : -1px;
    background      : url("top-left.gif");
}

span.bottom-left
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    bottom          : -1px;
    left            : -1px;
    background      : url("bottom-left.gif");
}

span.bottom-right
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    bottom          : -1px;
    right           : -1px;
    background      : url("bottom-right.gif");
}

span.top-right
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    top             : -1px;
    right           : -1px;
    background      : url("top-right.gif");
}
```

Javascript

```
function showHideSelect(select)
{
    var objSelect = document.getElementById(select);
    objSelect.style.display = (objSelect.style.display
== 'block') ? 'none' : 'block';
}

function validAndHide(txt, obj, input, select)
{
    document.getElementById(input).value = txt;
    obj.parentNode.parentNode.style.display = 'none';
    document.getElementById(select).innerHTML =
obj.innerHTML;
}
```

Voir le résultat : Lien21

Retrouvez l'article de Josselin Willette en ligne : Lien22

Bien débuter en .NET

Vous souhaitez débuter en .NET ? Vous ne savez pas par quoi commencer ? Ce guide va vous aider dans vos premiers pas, en passant en revue les différentes ressources utiles pour développer avec la plateforme .NET.

1. Qu'est-ce que la Plateforme .NET ?

La Plateforme .NET est un ensemble de composants technologiques de l'entreprise Microsoft. Ils permettent de bâtir des solutions métiers. Ils sont pour la plupart dépendant du framework .NET.

2. Qu'est-ce que le Framework .NET ?

Le Framework .NET est une technologie commune à la plateforme qui permet de rationaliser la collaboration entre les différents produits. Plusieurs langages sont disponibles comme le C#, le J# et le Visual Basic .NET.

La FAQ .NET sur Developpez.com : Lien23

3. Je débute

Les débuts ne sont jamais très aisés, notamment en ce qui concerne le choix des livres, la recherche de tutoriels en ligne et dans le choix des éditeurs ou EDI (Environnement de Développement Intégré). Les catégories ci-dessous vous permettront de vous acquitter de cette tâche souvent très longue et désagréable pour les débutants, tout ce que vous avez à faire, c'est de suivre ces différentes parties !

3.1. Quels tutoriels en ligne puis-je lire ?

- Les meilleurs cours pour débuter séléctionnés et écrits par la rédaction : Lien24
- Présentation du .NET Framework 2.0 : Lien25
- Introduction au langage C# : Lien26
- Cours d'initiation au VB.NET : Lien27
- Un cours complet en C# pour les débutants : Lien28

3.2. Quels livres puis-je lire ?

Les meilleurs livres séléctionnés par la rédaction (Lien29) et plus particulièrement :

- La Plateforme .NET
- La bible du développeur C# : Pratique de .NET 2 et C# 2

3.3. Quels sont les outils dont j'ai besoin ?

Pour développer des applications en .NET, vous aurez tout d'abord besoin du SDK du Framework .NET. Le plus répandu actuellement est le Framework 2.0. Vous pouvez tout aussi bien développer avec le dernier Framework de Microsoft à savoir le Framework .NET 3.0, dans ce cas là vous aurez besoin du SDK du Framework 3.0

Frameworks .NET

Tous les Framework .NET : Lien30

Pour développer il vous faut aussi un EDI (Environnement de Développement Intégré).Plusieurs EDI sont à votre disposition.
Les meilleurs EDI séléctionnés par la rédaction : Lien31

En gratuit sous Windows je vous conseille fortement Visual Studio 2005 Express Edition, qui n'est qu'une version allégée de Visual Studio mais qui n'en reste pas moins puissante.

Visual Studio 2005 Express Edition : Lien32
Les meilleurs tutoriels séléctionnés et écrits par la rédaction sur Visual Studio : Lien33

Vous pouvez aussi développer en .NET sous Linux. Et oui un portage est en cours de développement. Ce projet se nomme Mono (Lien34).

4. Je veux approfondir mes connaissances

Vous avez acquis un certain niveau avec le framework .NET, vous connaissez les bases du langage ainsi que les bases de la POO (Programmation Orientée Objet), et vous souhaitez passer au niveau supérieur ? Suivez le guide !

4.1. Quels tutoriels en ligne puis-je lire ?

Les meilleurs tutoriels avancés, séléctionnés et écrits par la rédaction :

- Les Windows Forms : Lien35
- L'ASP.NET : Lien36
- Le développement .NET sur PocketPC : Lien37
- .NET et les bases de données : Lien38
- Sharepoint : Lien39

4.2. Quels livres puis-je lire ?

- Les meilleurs livres sur l'ASP.NET séléctionnés par la rédaction : Lien40
- C# 2 de l'apprentissage du langage au développement ASP ... maîtrisez C# : Lien41
- Programmation mobile avec C# .NET : Lien42
- Programming Windows Workflow Foundation : Lien43

4.3. Quels outils supplémentaires sont à ma disposition ?

Bizarrement vous n'aurez pas forcément besoin d'outils supplémentaires, Visual Studio étant déjà bien assez complet.

Vous voulez réaliser des interfaces graphiques en .NET 2 ou développer ASP.NET ? Visual Studio intègre déjà ces fonctionnalités.

Vous voulez réaliser des interfaces graphiques en .NET 3 ? En attendant Visual Studio 2008 qui intégrera cette fonctionnalité.
Expression Blend (payant) : Lien44

Vous voulez travailler avec des bases de données SQL Server ? Si

vous avez installé SQL Server Express en même temps que Visual Studio vous n'avez besoin de rien d'autre. Dans le cas contraire il vous le faut.
SQL Server Express Edition: Lien45

Vous aurez sûrement aussi besoin d'un outil pour créer/modifer vos bases.
Microsoft SQL Server Management Studio Express : Lien46

Vous voulez développer en Silverlight ?
Les outils requis : Lien47

Cette liste n'est bien entendu pas exhaustive, il est possible de réaliser énormément de choses avec la plateforme .NET, c'est à vous de vous renseigner sur ce dont vous aurez besoin pour réaliser telle ou telle chose.

5. Où puis-je trouver des exercices à faire ?

Il n'est jamais très simple de trouver des exercices d'un niveau assez correct. Les liens ci-dessous vont vous amener vers des exercices pour vous entraîner.

- Le coach ASP.NET sur Microsoft.com : Lien48
- Le coach C# sur Microsoft.com : Lien49
- Le coach VB.NET sur Microsoft.com : Lien50
- Le coach VSTS sur Microsoft.com : Lien51
- Exercices en C# sur Developpez.com : Lien52

6. Je veux aller plus loin

Vous êtes à l'aise avec le Framework et la POO (Programmation Orientée Objet), vous avez une certaine maîtrise du langage et vous souhaitez passer à la vitesse supérieure encore une fois ? Par exemple développer en Silverlight ou créer de jeux. Jetez un oeil en dessous !

6.1. Je veux développer en Silverlight

La nouvelle technologie Internet de Microsoft vous intéresse ? Vous avez raison.
Le site officiel : Lien53
Silverlight 1.1 sur Developpez.com : Lien54
Réalisez un chat en Silverlight 1.1 sur Developpez.com : Lien55

6.2. Je veux créer des jeux

6.2.1. En 2D

La plateforme en vogue pour créer des jeux 2D (3D aussi) est XNA (XNA Is Not Acronymed), une plateforme de chez Microsoft, pour créer des jeux aussi bien pour PC que pour XBox 360.
XNA sur Microsoft.com : Lien56
XNA Creators Club : Lien57

Voici quelques tutoriels par la rédaction :

- Présentation de la plateforme XNA : Lien58
- Programmation XNA : Lien59

Outre la plateforme de Microsoft, vous pouvez trouver d'autres bibliothèques pour développer des jeux 2D.
Principalement il y a la très célèbre SDL qui possède un wrapper en .NET (C# et VB.NET).
SDL.NET : Lien60
Tutoriels SDL.NET : Lien61

6.2.2. En 3D

Du côté de la 3D, on a aussi XNA, qui permet de réaliser des jeux 3D. Cette plateforme est assez bas-niveau (plus que DirectX mais moins qu'un moteur 3D).

Quelques tutoriels pour la 3D avec XNA :

- Riemers XNA Tutorials : Lien62
- XNA Creators Club : Samples : Lien63
- Ziggyware : Lien64

Outre la plateforme XNA, vous pouvez trouver moultes moteurs 3D soit écrits en .NET, soit possédant un wrapper .NET.

nxEngine est un moteur entièrement écrit en C# par funkydata. Ce moteur est très prometteur.
Site officiel sur Developpez.com : Lien65
Le forum sur Developpez.net : Lien66

Brume est un moteur lui aussi entièrement écrit en C#. Il est lui aussi très prometteur.
Brume Game Engine : Lien67

Ensuite nous avons les moteurs possédant un wrapper .NET.

Les plus connus :

- Ogre.NET : Lien68
- Irrlicht.NET : Lien69

7. J'ai tout lu mais il y a des choses que je n'ai pas comprises, que faire ?

Pas de panique ! Vous pouvez toujours poser vos questions sur les forums de Developpez.com !

Les Forums :

- Les forums Dotnet : Lien70
- Forum : Développement 2D, 3D et Jeux : Lien71
- Forum : SQL Server : Lien72
- Forum : Sharepoint : Lien73

Bien sûr, il ne faut pas non plus oublier d'aller jeter un oeil aux FAQ's de Developpez.com, qui rassemblent un grand nombres de Questions/Réponses.

- Les FAQ's Dotnet : Lien74
- FAQ : SQL Server : Lien75
- FAQ : Sharepoint : Lien76

8. Conclusion

J'espère que ce guide vous aura permis d'y voir un peu plus clair et de commencer votre apprentissage de la plateforme .NET plus sereinement.
L'équipe .NET vous souhaite un bon développement, en espérant vous voir sur nos forums, et pourquoi pas dans quelques temps aider sur ces mêmes forums ou intégrer l'équipe.

Retrouvez l'article de Benjamin Roux en ligne : Lien77

Les derniers tutoriels et articles

Initiation à la programmation multitâche en C avec Pthreads

Les threads permettent de créer des programmes multitâches, ce tutoriel vous propose une approche par la pratique en partant d'un exemple unique !

1. Introduction

La programmation multitâche a toujours été et est toujours, un sujet assez complexe, essentiellement dû au manque de documentation et tutoriels en français !

Ce tutoriel ne se veut pas une documentation complète (il existe des ouvrages dédiés sur ce sujet) mais plutôt une initiation pour vous donner un aperçu de ce qu'il est possible de faire et nous ferons un petit tour d'horizon de la bibliothèque pthread au fil de l'eau.

Nous nous baserons sur un sujet d'exemple unique soit, une gestion correcte d'un stock de magasin et des clients. Le but bien entendu, est que le stock ne descende jamais en-dessous de zéro, ce qui dans la réalité n'est pas faisable donc, si le stock descend à zéro ou si le client courant demande plus de produit qu'il y en a en stock, il faut renflouer celui-ci. Chaque client est représenté par un thread, dans notre exemple nous allons en créer 5, la gestion du stock est également un thread supplémentaire.

2. Avant de commencer

La bibliothèque Pthreads est portable, elle existe sur Linux ainsi que sur Windows. Si vous êtes sur Windows, il vous faudra cependant l'installer car elle ne l'est pas d'office sur ce système ! Vous pouvez télécharger la bibliothèque sur le site suivant: Lien78.

Pour pouvoir compiler un projet (tous systèmes) avec pthread, il faut pour commencer, ajouter l'en-tête :

```
#include <pthread.h>
```

ainsi qu'ajouter à l'éditeur de lien la bibliothèque :

```
-lpthread
```

et spécifier au compilateur la constante :

```
-D_REENTRANT
```

Vous voilà prêt pour compiler des programmes utilisant la bibliothèque Pthreads !

Attention, avec le compilateur MingW sous Windows, si vous développez une application C++ utilisant des exceptions, il est nécessaire de compiler et de réaliser l'édition des liens avec l'option -mthreads. En effet, d'origine, les exceptions ne sont pas thread-safe, l'option mthreads permet qu'elles le soient.

Les utilisateurs de Linux devront spécifier dans la ligne de compilation des exemples de ce tutoriel, la constante -DLinux et les utilisateurs de Windows -DWin32. Cela sert à prendre en charge de façon portable la mise en pause des portions du code d'exemple. Un fichier Makefile est également disponible pour les utilisateurs de Linux !

3. Qu'est-ce qu'un thread ?

Un thread (Fil ou encore Fil d'exécution) est une portion de code (fonction) qui se déroule en parallèle au thread principal (aussi appelé main). Ce principe est un peu semblable à la fonction fork sur Linux par exemple sauf que nous ne faisons pas de copie du processus père, nous définissons des fonctions qui vont se lancer en même temps que le processus, ce qui permet de faire de la programmation multitâche. Le but est donc de permettre au programme de réaliser plusieurs actions au même moment (imaginez un programme qui fait un gros calcul et une barre de progression qui avance en même temps).

On peut également considérer un thread comme un processus allégé pour mieux imager le tout ! En comparaison des threads, un fork prend en moyenne 30 fois plus de temps à faire !

4. Création et exécution des threads

4.1. pthread_create

Un thread se créé avec la fonction :

```
int pthread_create (pthread_t * thread, pthread_attr_t
* attr, void * (* start_routine) (void *), void * arg);
```

Voyons dans l'ordre, à quoi correspondent ses arguments :

1. Le type **pthread_t** est un type opaque, sa valeur réelle dépend de l'implémentation (sur Linux il s'agit en générale du type **unsigned long**). Ce type correspond à l'identifiant du thread qui sera créé, tout comme les processus on leur propre identifiant.
2. Le type **pthread_attr_t** est un autre type opaque permettant de définir des attributs spécifiques pour chaque thread mais cela dépasse le cadre de ce tutoriel. Il faut simplement savoir que l'on peut changer le comportement de la gestion des threads comme par exemple, les régler pour qu'ils tournent sur un système temps réel ! En générale on se contente des attributs par défaut donc en mettant cet argument à NULL.
3. Chaque thread dispose d'une fonction à exécuter, c'est en même temps sa raison de vivre... Cet argument permet de transmettre un pointeur sur la fonction qu'il devra exécuter.
4. Ce dernier argument représente un argument que l'on peut passer à la fonction que le thread doit exécuter.

Si la création réussit, la fonction renvoie **0** (zéro) et l'identifiant du thread nouvellement créé est stocké à l'adresse fournie en premier argument. En cas d'erreur, la valeur **EAGAIN** est retournée par la fonction s'il n'y a pas assez de ressources système pour créer un nouveau thread ou bien si le nombre maximum de threads définit par la constante **PTHREAD_THREADS_MAX** est atteint !

Le nombre de threads simultanés est limité suivant les systèmes. La constante **PTHREAD_THREADS_MAX** définie le nombre maximum qui est de **1024** sur les Unixoïdes !

Lorsque le thread est créé, il est lancé immédiatement et exécute la fonction passée en troisième argument. L'exécution du thread se fait soit jusqu'à la fin de sa fonction ou bien jusqu'à son annulation, c'est ce que nous allons voir au prochain chapitre.

Il est possible d'attribuer la même fonction à plusieurs threads !

5. Annulation et fin des threads

5.1. pthread_exit

On peut arrêter le thread courant avec la fonction :

```
void pthread_exit (void * retval);
```

Son seul argument est le retour de la fonction du thread appelant. Cet argument peut aussi être récupéré par la fonction pthread_join que nous verrons plus bas.

5.2. pthread_cancel

On peut annuler un thread à partir d'un autre à n'importe quel moment avec la fonction :
int pthread_cancel (pthread_t thread);

L'argument de cette fonction est le thread à annuler. Elle renvoie **0** (zéro) si elle réussie ou la valeur **ESRCH** si aucun thread ne correspond à celui passé en argument.

Il faut cependant être très vigilant lors de l'utilisation de cette fonction. En effet, si le thread dont on demande l'arrêt possède un verrou et ne l'a toujours pas relâché, il y a un risque de laisser ce verrou dans l'état verrouillé, il sera alors dans ce cas impossible de le récupérer !

Pour éviter ce genre de phénomène, on peut avoir recours à une fonction permettant de changer le comportement du thread par rapport aux requêtes d'annulations, ce que nous allons voir ci-dessous !

5.2.1. pthread_setcancelstate

```
int pthread_setcancelstate (int state, int *
etat_pred);
```

Cette fonction permet de changer le comportement du thread appelant par rapport aux requêtes d'annulations. Ces arguments sont dans l'ordre:

- Etat d'annulation. Il peut prendre les deux valeurs suivantes :
  - **PTHREAD_CANCEL_ENABLE** : Autorise les annulations pour le thread appelant.
  - **PTHREAD_CANCEL_DISABLE** : Désactive les requêtes d'annulation.
- Adresse vers l'état précédent (ou NULL) permettant ainsi sa restauration lors d'un prochain appel de la fonction.

La fonction renvoie la valeur **0** en cas de succès ou **EINVAL** si l'argument ne correspond ni à PTHREAD_CANCEL_ENABLE et ni à PTHREAD_CANCEL_DISABLE !

5.3. pthread_join

Lorsque nous créons des threads puis nous laissons continuer par exemple la fonction main, nous prenons le risque de terminer le programme complètement sans avoir pu exécuter les threads. Nous devons en effet attendre que les différents threads créés se terminent. Pour cela, il existe la fonction :

```
int pthread_join (pthread_t th, void ** thread_return);
```

Ses arguments sont dans l'ordre :

1. Le thread à attendre.
2. La valeur de retour de la fonction du thread **th**.

L'appel de cette fonction met en pause l'exécution du thread appelant jusqu'au retour de la fonction. Si aucun problème n'a eu lieu, elle retourne 0 (zéro) et la valeur de retour du thread est passé à l'adresse indiquée (second argument) si elle est différente de NULL. En cas de problème, la fonction retourne une des valeurs suivantes :

- **ESRCH** : Aucun thread ne correspond à celui passé en argument.
- **EINVAL** : Le thread a été détaché ou un autre thread attend déjà la fin du même thread.
- **EDEADLK** : Le thread passé en argument correspond au thread appelant.

Un thread terminé ne peut être relancé, il faut en créer un nouveau car un thread qui touche à sa fin est implicitement détruit !

6. Mise en pratique

Dans ce programme, nous créons 1 thread pour la gestion du stock du magasin et 5 threads pour les clients. Les deux fonctions fn_store et fn_clients sont des boucles infinies (pour cet exemple mais dans la réalité ca ne sera pas toujours le cas) exécutant les mêmes tâches. Les threads clients prennent dans le stock et le thread du magasin va renflouer le stock dès qu'il devient trop bas pour satisfaire les clients. Le nombre d'articles pris du stock sont des nombres aléatoires ainsi que l'ordre de passage des clients.

Nous pouvons voir qu'à la fin de la fonction main, nous avons une boucle qui parcourt chaque threads en lançant la fonction pthread_join. Ceci permet d'attendre la fin des threads et évite donc que le programme ne se termine et quitte prématurément les threads !

6.1. Code complet

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

#if defined (Win32)
#  include <windows.h>
#  define psleep(sec) Sleep ((sec) * 1000)
#elif defined (Linux)
#  include <unistd.h>
#  define psleep(sec) sleep ((sec))
#endif

#define INITIAL_STOCK   20
#define NB_CLIENTS      5
```

```
/* Structure stockant les informations des threads
clients et du magasin. */
typedef struct{
   int stock;

   pthread_t thread_store;
   pthread_t thread_clients [NB_CLIENTS];
}
store_t;

static store_t store =
{
   .stock = INITIAL_STOCK,
};

/* Fonction pour tirer un nombre au sort entre 0 et
max. */
static int get_random (int max){
   double val;

   val = (double) max * rand ();
   val = val / (RAND_MAX + 1.0);

   return ((int) val);
}

/* Fonction pour le thread du magasin. */
static void * fn_store (void * p_data){
   while (1){
      if (store.stock <= 0){
         store.stock = INITIAL_STOCK;
         printf ("Remplissage du stock de %d articles !
\n", store.stock);
      }
   }

   return NULL;
}

/* Fonction pour les threads des clients. */
static void * fn_clients (void * p_data){
   int nb = (int) p_data;

   while (1){
      int val = get_random (6);

      psleep (get_random (3));

      store.stock = store.stock - val;
      printf (
         "Client %d prend %d du stock, reste %d en
stock !\n",
         nb, val, store.stock
      );
   }

   return NULL;
}

int main (void){
   int i = 0;
   int ret = 0;

   /* Creation du thread du magasin. */
   printf ("Creation du thread du magasin !\n");
   ret = pthread_create (
      & store.thread_store, NULL,
      fn_store, NULL
   );

   /* Creation des threads des clients si celui du
magasin a reussi. */
   if (! ret){
      printf ("Creation des threads clients !\n");
      for (i = 0; i < NB_CLIENTS; i++){
         ret = pthread_create (
            & store.thread_clients [i], NULL,
            fn_clients, (void *) i
         );

         if (ret){
            fprintf (stderr, "%s", strerror (ret));
         }
      }
   }
   else{
      fprintf (stderr, "%s", strerror (ret));
   }

   /* Attente de la fin des threads. */
   i = 0;
   for (i = 0; i < NB_CLIENTS; i++){
      pthread_join (store.thread_clients [i], NULL);
   }
   pthread_join (store.thread_store, NULL);

   return EXIT_SUCCESS;
}
```

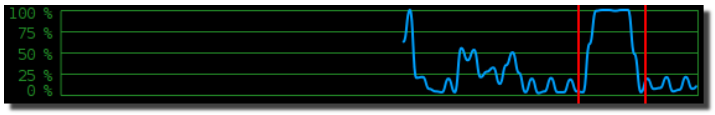
L'exemple de code ci-dessus utilise une variable globale ! Ici ce n'est qu'à titre d'exemple mais je vous encourage à éviter ce genre de pratique autant que possible !

6.2. Sortie du programme

Voici la sortie du programme sur la console avec annulation utilisateur :

```
Creation du thread du magasin !
Creation des threads clients !
Client 2 prend 5 du stock, reste 15 en stock !
Client 0 prend 5 du stock, reste 10 en stock !
Client 3 prend 1 du stock, reste 9 en stock !
Client 4 prend 2 du stock, reste 7 en stock !
Client 1 prend 4 du stock, reste 3 en stock !
Client 2 prend 2 du stock, reste 1 en stock !
Client 2 prend 0 du stock, reste 1 en stock !
Client 0 prend 2 du stock, reste -1 en stock !
Remplissage du stock de 20 articles !
Client 1 prend 0 du stock, reste 20 en stock !
Client 1 prend 0 du stock, reste 20 en stock !
Client 1 prend 5 du stock, reste 15 en stock !
Client 0 prend 0 du stock, reste 15 en stock !
Client 0 prend 3 du stock, reste 12 en stock !
Client 3 prend 5 du stock, reste 7 en stock !
Client 4 prend 3 du stock, reste 4 en stock !
Client 2 prend 0 du stock, reste 4 en stock !
Client 0 prend 3 du stock, reste 1 en stock !
Client 1 prend 3 du stock, reste -2 en stock !
Client 3 prend 2 du stock, reste -4 en stock !
Client 4 prend 1 du stock, reste -5 en stock !
Remplissage du stock de 20 articles !
Client 2 prend 3 du stock, reste 17 en stock !
Client 2 prend 3 du stock, reste 14 en stock !
Client 1 prend 1 du stock, reste 13 en stock !
Client 0 prend 2 du stock, reste 11 en stock !
Client 0 prend 0 du stock, reste 11 en stock !
Client 0 prend 2 du stock, reste 9 en stock !

<CTRL-C>
```

Et la charge CPU utilisée durant le déroulement du processus et ses threads :



6.3. Observations

Nous pouvons voir grâce à ces données crées lors du déroulement du programme que les clients se servent même si plus aucun produit n'est en stock, le stock est donc durant un court moment en négatif, ce qu'il faut éviter dans la réalité !

Non seulement les clients se servent au petit bonheur mais en plus, sans prendre en considération que d'autres clients peuvent également prendre dans le même stock et bien sûr, sans prendre en compte que le magasin peut renflouer son stock quand il est au plus bas.

Ceci pour noter une chose importante : Les threads ne tiennent en aucun cas compte qu'une autre fonction puisse également accéder à la même variable (accès concurrentiels), en lecture mais aussi en écriture. Cela peut avoir des répercutions dramatiques dans une application critique !

Le prochain chapitre va donc aborder la protection pour les accès concurrentiels autrement dit, les mutex !

En outre, nous pouvons observer la charge CPU utilisée lors du déroulement du programme. On peut noter que les ressources sont utilisées au maximum, ceci surtout dû au fait que la fonction fn_store tourne sans cesse jusqu'à ce qu'on mette fin au programme ! Nous aborderons ce sujet dans la dernière partie de ce tutoriel.

7. Les mutex

Le(s) **mutex** (mutual exclusion ou zone d'exclusion mutuelle), est(sont) un système de verrou donnant ainsi une garantie sur la viabilité des données manipulées par les threads. En effet, il arrive même très souvent que plusieurs threads doivent accéder en lecture et/ou en écriture aux mêmes variables. Si un thread possède le verrou, seulement celui-ci peut lire et écrire sur les variables étant dans la portion de code protégée (aussi appelée zone critique). Lorsque le thread a terminé, il libère le verrou et un autre thread peut le prendre à son tour.

Pour créer un mutex, il faut tout simplement déclarer une variable du type **pthread_mutex_t** et l'initialiser avec la constante **PTHREAD_MUTEX_INITIALIZER** soit par exemple :

```
static pthread_mutex_t mutex_stock =
PTHREAD_MUTEX_INITIALIZER;
```

Un mutex n'a que deux états possibles, il est soit verrouillé soit déverrouillé. On utilise les deux fonctions ci-dessous pour changer les états.

7.1. pthread_mutex_lock

```
int pthread_mutex_lock (pthread_mutex_t * mutex);
```

Cette fonction permet de déterminer le début d'une zone critique. Son seul argument est l'adresse d'un mutex de type pthread_mutex_t. La fonction renvoie 0 en cas de succès ou l'une des valeurs suivante en cas d'échec:

- **EINVAL** : mutex non initialisé.
- **EDEADLK** : mutex déjà verrouillé par un thread différent.

7.2. pthread_mutex_unlock

int pthread_mutex_unlock (pthread_mutex_t * mutex);

Cette fonction permet de relâcher le verrou passé en argument qui est l'adresse d'un mutex de type pthread_mutex_t. La fonction renvoie 0 en cas de succès ou l'une des valeurs suivante en cas d'échec:

- **EINVAL** : mutex non initialisé.
- **EPERM** : le thread n'a pas la main sur le mutex.

8. Mise en pratique

Dans la seconde version de notre exemple, des zones critiques ont étés définies dans les fonctions fn_store et fn_clients. On peut remarquer que nous prenons et libérons le mutex à chaque tour de boucle ce qui permet de ne pas bloquer le programme et ainsi, chaque thread aura l'opportunité de le prendre à son tour pour accomplir sa tâche.

Avant chaque arrêt/annulation/fin d'un thread, il ne faut surtout pas oublier de libérer les verrous car vous risquez le cas échéant, d'obtenir ce qu'on appelle un Dead Lock, le mutex est verrouillé et le restera. Tous les threads voulant l'utiliser vont s'arrêter.

8.1. Code complet

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

#if defined (Win32)
#  include <windows.h>
#  define psleep(sec) Sleep ((sec) * 1000)
#elif defined (Linux)
#  include <unistd.h>
#  define psleep(sec) sleep ((sec))
#endif

#define INITIAL_STOCK   20
#define NB_CLIENTS      5

/* Structure stockant les informations des threads
clients et du magasin. */
typedef struct{
   int stock;

   pthread_t thread_store;
   pthread_t thread_clients [NB_CLIENTS];

   pthread_mutex_t mutex_stock;
}
store_t;

static store_t store ={
   .stock = INITIAL_STOCK,
   .mutex_stock = PTHREAD_MUTEX_INITIALIZER,
};

/* Fonction pour tirer un nombre au sort entre 0 et
max. */
static int get_random (int max){
   double val;

   val = (double) max * rand ();
   val = val / (RAND_MAX + 1.0);

   return ((int) val);
}
```

```
/* Fonction pour le thread du magasin. */
static void * fn_store (void * p_data){
   while (1){
      /* Debut de la zone protegee. */
      pthread_mutex_lock (& store.mutex_stock);

      if (store.stock <= 0){
         store.stock = INITIAL_STOCK;
         printf ("Remplissage du stock de %d articles !
\n", store.stock);
      }

      /* Fin de la zone protegee. */
      pthread_mutex_unlock (& store.mutex_stock);
   }

   return NULL;
}

/* Fonction pour les threads des clients. */
static void * fn_clients (void * p_data){
   int nb = (int) p_data;

   while (1){
      int val = get_random (6);

      /* Debut de la zone protegee. */
      pthread_mutex_lock (& store.mutex_stock);

      psleep (get_random (3));

      store.stock = store.stock - val;
      printf (
         "Client %d prend %d du stock, reste %d en
stock !\n",
         nb, val, store.stock
      );

      /* Fin de la zone protegee. */
      pthread_mutex_unlock (& store.mutex_stock);
   }

   return NULL;
}

int main (void){
   int i = 0;
   int ret = 0;

   /* Creation du thread du magasin. */
   printf ("Creation du thread du magasin !\n");
   ret = pthread_create (
      & store.thread_store, NULL,
      fn_store, NULL
   );

   /* Creation des threads des clients si celui du
magasinn a reussi. */
   if (! ret){
      printf ("Creation des threads clients !\n");
      for (i = 0; i < NB_CLIENTS; i++){
         ret = pthread_create (
            & store.thread_clients [i], NULL,
            fn_clients, (void *) i
         );

         if (ret){
            fprintf (stderr, "%s", strerror (ret));
         }
      }
   }
   else{
      fprintf (stderr, "%s", strerror (ret));
   }


   /* Attente de la fin des threads. */
   i = 0;
   for (i = 0; i < NB_CLIENTS; i++){
      pthread_join (store.thread_clients [i], NULL);
   }
   pthread_join (store.thread_store, NULL);

   return EXIT_SUCCESS;
}
```

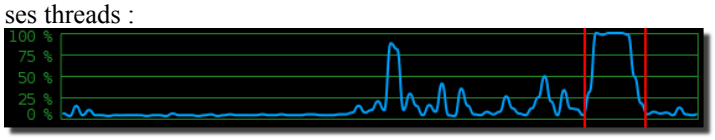
L'exemple de code ci-dessus utilise une variable globale ! Ici ce n'est qu'à titre d'exemple mais je vous encourage à éviter ce genre de pratique autant que possible !

8.2. Sortie du programme

Voici la sortie du programme sur la console avec annulation utilisateur :

```
Creation du thread du magasin !
Creation des threads clients !
Client 2 prend 5 du stock, reste 15 en stock !
Client 0 prend 5 du stock, reste 10 en stock !
Client 3 prend 1 du stock, reste 9 en stock !
Client 4 prend 2 du stock, reste 7 en stock !
Client 1 prend 4 du stock, reste 3 en stock !
Client 2 prend 2 du stock, reste 1 en stock !
Client 2 prend 0 du stock, reste 1 en stock !
Client 0 prend 2 du stock, reste -1 en stock !
Remplissage du stock de 20 articles !
Client 1 prend 0 du stock, reste 20 en stock !
Client 1 prend 0 du stock, reste 20 en stock !
Client 1 prend 5 du stock, reste 15 en stock !
Client 3 prend 5 du stock, reste 10 en stock !
Client 3 prend 3 du stock, reste 7 en stock !
Client 4 prend 3 du stock, reste 4 en stock !
Client 0 prend 0 du stock, reste 4 en stock !
Client 2 prend 0 du stock, reste 4 en stock !
Client 3 prend 3 du stock, reste 1 en stock !
Client 1 prend 3 du stock, reste -2 en stock !
Client 4 prend 2 du stock, reste -4 en stock !
Client 0 prend 1 du stock, reste -5 en stock !
Remplissage du stock de 20 articles !
Client 2 prend 3 du stock, reste 17 en stock !
Client 2 prend 3 du stock, reste 14 en stock !
Client 3 prend 2 du stock, reste 12 en stock !
Client 1 prend 1 du stock, reste 11 en stock !
Client 1 prend 0 du stock, reste 11 en stock !
Client 1 prend 2 du stock, reste 9 en stock !

<CTRL-C>
```

Et la charge CPU utilisée durant le déroulement du processus et ses threads :



8.3. Observations

Il n'y a pas à vrai dire, de changement radical par rapport à l'affichage du résultat sur la sortie console mais nous avons néanmoins protégé l'accès aux données, ce qui fait un risque en moins. Bien sûr, ce programme d'exemple n'est pas un programme critique mais sur d'autres applications cela peut avoir des effets

désastreux !

Nous pouvons également remarquer que la charge CPU reste inchangée, le programme prend presque toutes les ressources disponibles du processeur. Pour pallier à ce problème, nous pouvons mettre des threads en attente jusqu'à ce que des conditions de réveil soient remplies ! Nous allons donc étudier dans la partie suivante, les conditions !

Retrouvez la suite de l'article de Frank Hecht en ligne : Lien79

FAQ Qt

Quelle est la licence d'utilisation de Qt4 ?

Depuis la version majeure 4.0.0, Qt4 est ditribué en double licence, commerciale et libre, sur les trois plateformes majeures, Windows, Linux et OSX.

La licence libre utilisée est la licence libre GPL2, et tout code développé avec la version GPL de Qt doit aussi être GPL. Pour la version commerciale, le plus simple est de suivre les liens sur le site de Trolltech.

Où trouver la documentation de Qt ?

Trolltech propose une documentation plus qu'exhaustive pour chacune des versions de Qt. Le sommaire de toutes ces documentations se trouve ici : Online Reference Documentation (Lien80) La documentation de la dernière version au moment où j'écris, Qt 4.3, se trouve à cet endroit : Lien81. Les documentations des futures versions seront disponibles depuis le sommaire indiqué plus haut.

Est-ce compliqué d'utiliser Qt 4 ?

En réalité, utiliser Qt est plus simple que ce que l'on pense. En effet, le code suivant affiche un bouton qui ferme l'application lorsque l'on clique dessus, et pourtant il ne fait que 13 lignes.

```
#include <QApplication> // Nécessaire pour créer une application avec Qt
#include <QPushButton> // Nécessaire pour pouvoir créer un bouton

int main(int argc, char **argv) // La fonction main() habituelle
{
    QApplication    app(argc, argv); // Qt récupère les arguments passés au programme
    QPushButton     quit("Hello World!"); // on crée notre bouton, intitulé "Hello World"
    quit.resize(300, 40); // on le redimensionne
    quit.setFont(QFont("Arial", 18, QFont::Bold)); // on change la police et la taille
    QObject::connect(&quit, SIGNAL(clicked()), &app, SLOT(quit())); // on explique à Qt que l'application doit se terminer lors du clic sur le bouton créé précédemment
    quit.show(); // on affiche le bouton
    return app.exec(); // on laisse Qt gérer le code de retour du programme
}
```

Comme vous pouvez le voir, il est très simple de gérer ses composants. Pour en découvrir plus, il est bon de consulter la documentation et les exemples de Trolltech, ainsi que les tutoriaux présents sur developpez.

Comment compiler des projets utilisant Qt 4 ?

Les développeurs de Qt 4 proposent des outils très utiles qui facilitent la gestion de vos projets utilisant Qt 4. En effet, il existe un outil permettant de transformer les fichiers du designer (.ui) en fichiers C++, un autre permettant de générer le code nécessaire pour la création de widgets personnalisés, ...

Cependant, ils fournissent également un outil permettant de gérer automatiquement les fichiers du designer, les fichiers dans lesquels vous définissez des widgets personnalisés, et même en réalité tout votre projet : qmake. Cet outil permet de générer un fichier de projet .pro, de générer à partir de ce dernier les règles de compilation de votre projet, et bien d'autres choses comme la détection de votre compilateur, du répertoire d'installation de Qt 4, ...

Un tutoriel a été écrit pour présenter cet outil et décrire son utilisation : Compilation des projets Qt 4 (Lien82).

Retrouvez ces Q/R sur la FAQ Qt : Lien83

Les derniers tutoriels et articles

Les variables temporaires dans Microsoft Access 2007

Découvrez les variables temporaires de Microsoft Access 2007

1. Introduction

Les variables temporaires font parties des nombreuses nouveautés apparues avec la version 2007 de Microsoft Access. Si leur intérêt est parfois négligé par bon nombre de développeurs, je les trouve, pour ma part, particulièrement pratiques notamment lors d'échange de données entre différents applicatifs.

En détail, il s'agit en fait de variables disponibles dans n'importe quel module d'une application (un peu comme les variables déclarées en Public) et qui offrent l'avantage d'être aussi accessibles depuis un autre projet via automation par exemple. Bien entendu, comme toute porte vers l'extérieur cela représente une faille de sécurité et ne doit être réservé qu'à des échanges et des stockages non critiques.

2. Manipulation

Les variables temporaires **TempVar** sont des objets regroupés au sein d'une collection **Application.TempVars** qui vous permet par conséquent de créer autant de variables temporaires que nécessaire sans toutefois dépasser la limite de 256. Attention cependant, il ne faut pas que l'utilisation des variables temporaires devienne une solution de facilité et systématique sans quoi vous risquez de vous retrouver avec des dizaines de variables inutiles transformant votre développement en usine à gaz.

La collection **TempVars** permet d'ajouter, de modifier et de supprimer des variables temporaires. A l'ouverture de l'application, aucune variable n'est disponible (Access n'en utilise pas par défaut, il s'agit d'une fonctionnalité réservée au développeur).

Chaque variable possède un nom et une valeur de type **Variant**. L'ajout est réalisé en invoquant la méthode **Add** de la collection.

Exemple :

```
Sub ajout()
    'Crée une nouvelle variable
    Application.TempVars.Add "Ma Variable", 100
    'Affiche la variable
    MsgBox Application.TempVars("Ma Variable").Value
End Sub
```

Ici, le nom de la variable est Ma Variable et sa valeur est 100. Comme vous pouvez le constater, l'accès à la variable se fait ensuite comme pour toute collection, c'est-à-dire en spécifiant directement le nom de l'objet initialement défini.

Il est aussi possible d'utiliser l'index de l'objet **TempVar** dans la collection.

Exemple :

```
MsgBox Application.TempVars(0).Value
```

Pour modifier le contenu d'une variable temporaire deux techniques peuvent être utilisées :

1. Invoquer la méthode Add de nouveau pour écraser l'ancienne valeur
2. Modifier la propriété Value de l'objet TempVar concerné

Premier exemple :

```
'Crée une nouvelle variable nommée Ma Variable
Application.TempVars.Add "Ma Variable", 100
'Ecrase la variable pour modifier son contenu
Application.TempVars.Add "Ma Variable", 105
'Affiche la variable
MsgBox Application.TempVars("Ma Variable").Value
```

Deuxième exemple :

```
'Crée une nouvelle variable nommée Ma Variable
Application.TempVars.Add "Ma Variable", 100
'Modifie la propriété Value
Application.TempVars("Ma Variable").Value = 105
'Affiche la variable
MsgBox Application.TempVars("Ma Variable").Value
```

Je vous recommande d'utiliser toujours la deuxième méthode afin d'éviter d'écraser des variables par mégarde et de perdre ainsi du temps en déboguage.

La méthode **Remove** permet de supprimer une variable temporaire en spécifiant son nom (ou son index) en argument. La méthode RemoveAll quant à elle permet de supprimer l'ensemble des variables temporaires.

```
'Crée une nouvelle variable nommée Ma Variable
Application.TempVars.Add "Ma Variable", 100
'Supprime la variable
Application.TempVars.Remove "Ma Variable"
```

Attention, si vous tentez de supprimer une variable qui n'existe pas, aucune erreur ne sera générée. De la même façon, si vous essayez d'accéder à une variable inexistante la valeur retournée sera **Null**. Cette trop grande permissivité est à regretter et donne un peu l'impression d'une fonctionnalité mal finie.

Enfin, comme pour toute collection, la propriété **Count** permet de connaître le nombre de variables temporaires utilisées :

```
Dim i As Integer
For i = 1 To 10
    Application.TempVars.Add "MaVariable" & i, i
Next i
MsgBox Application.TempVars.Count
```

2.1. En résumé

Add	**Méthode**. Ajoute la variable temporaire dont le nom et la valeur sont passés en paramètre.
Count	**Propriété (Long)**. Retourne le nombre de variables temporaires créées.
Item	**Méthode**. Retourne l'objet **TempVar** correspondant dont le nom ou l'indice est passé en paramètre.
Remove	**Méthode**. Supprime la variable temporaire dont le nom ou l'indice est passé en paramètre.
RemoveAll	**Méthode**. Supprime toutes les variables temporaires.

3. Utilisation des variables temporaires dans un formulaire

Il est possible d'afficher directement la valeur d'une variable temporaire dans un contrôle en plaçant la syntaxe d'appel dans la propriété **Source** du contrôle concerné.

Par exemple, dans le cadre d'une zone de texte affichant l'heure à laquelle l'application a été ouverte :

1. La macro **Autoexec** renseigne la variable temporaire **tmpDateHeure** avec le code VBA suivant :

```
Function AutoExec()
    Application.TempVars.Add "tmpDateHeure",
Format(Now, "dd mmmm yyyy à hh:nn:ss")
End Function
```

2. Le formulaire possède une zone de texte dont la **source** est :

```
=[Application].[VarTemp]("tmpDateHeure")
```

Le générateur d'expression traduit Application.TempVars en [Application].[VarTemp] automatiquement.

4. Utilisation des variables temporaires dans une requête

Si vous souhaitez utiliser la valeur d'une variable temporaire dans une requête, vous ne pouvez pas utiliser une syntaxe similaire à celle vue juste avant. Vous devez obligatoirement passer par une fonction VBA qui sera chargée de retourner le contenu de la variable passée en paramètre. Vous intégrerez ensuite cette fonction dans votre SQL.

```
Function RetourTempVar(strTempVar As String) As Variant
    RetourTempVar = Application.TempVars(strTempVar)
End Function
```

Le code SQL qui aurait pu être :

```
SELECT [Application].[VarTemp]("tmpDateHeure")
FROM MaTable
```

Devient :

```
SELECT RetourTempVar("tmpDateHeure")
FROM MaTable
```

Pour les habitués, il s'agit d'un procédé identique à celui employé pour utiliser la fonction Replace dans les requêtes d'Access 2000

5. Mise en évidence dans le cadre d'automation

Pour illustrer l'échange de données entre deux applications à l'aide des variables temporaires nous allons créer deux fichiers Microsoft Access. Le premier se contentera d'afficher l'état d'une variable temporaire, le second sera chargé de modifier régulièrement les variables temporaires du précédent et de lancer l'affichage (l'API Windows Sleep permettra de temporiser).

5.1. Client.accdb

Créez une nouvelle base de données client.accdb et ajoutez-y un module nommé mduClient contenant le code suivant :

```
Option Compare Database

Function Afficher()
    With Application
        'Teste si la variable temporaire "Ma Variable"
existe
        If Not IsNull(.TempVars("Ma Variable").Value)
Then
            MsgBox .TempVars("Ma Variable").Value
        End If
    End With
End Function
```

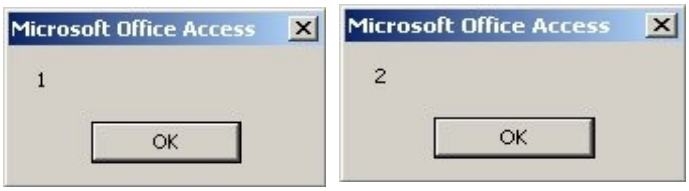
5.2. Serveur.accdb

Nous allons procéder de la même façon avec un module nommé mduServeur contenant le code suivant :

```
Option Compare Database

Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds
As Long)

Function Dialoguer()
'Déclararion de la variable Application correspondant
à Client.accdb
Dim oCliApp As Access.Application
'Crée une nouvelle instance d'Access
Set oCliApp = New Access.Application
'Ouvre le fichier client.accdb
oCliApp.OpenCurrentDatabase ("D:\client.accdb")

With oCliApp
    'Crée la variable temporaire
    .TempVars.Add "Ma Variable", 0

    'Toutes les 2 secondes, le programme va modifier la
variable temporaire
    'Ma Variable dans client.accdb
    While True
        .TempVars("Ma Variable") = .TempVars("Ma
Variable") + 1
        oCliApp.Run "Afficher"
        DoEvents
        Sleep 2000
    Wend
```

```
End With
End Function
```

Résultats :



6. Programmation Orientée Objet

Jusque là, nous avons utilisé des variables numériques. La propriété Value d'un objet TempVar étant Variant, il est en fait possible de stocker n'importe quelle variable d'un sous-type de Variant : Integer, Currency, String, Single, Boolean, etc. (y compris la valeur NULL). En revanche si vous tentez d'y stocker un objet vous rencontrer l'erreur 32538 : Les variables temporaires ne peuvent contenir que des données, pas des objets.

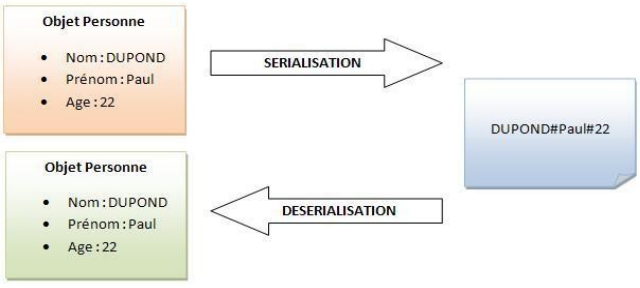
Alors comment procéder pour stocker un objet ? Bien souvent ce qui est intéressant de conserver ou d'échanger dans un objet ne sont pas ses méthodes (facilement réutilisable depuis un autre objet de la même classe) mais plutôt la valeur de ses propriétés. Pour cela deux cas de figures :

- Stocker chaque propriété dans un objet TempVar correspondant. Cette solution peut s'avérer lourde dans le cas d'objets proposant un grand nombre de propriétés.
- Encoder l'ensemble des propriétés dans un objet TempVar unique pour cet objet.

Cette deuxième méthode se rapproche du mécanisme de sérialisation que l'on retrouve dans d'autres langages tels que Java. Malheureusement une telle fonctionnalité n'est pas native dans Visual Basic et va demander à être codée par le développeur en fonction de ses besoins et ne pourra par conséquent se limiter qu'aux objets simples dont les propriétés sont des données et non d'autres objets.

Ce mécanisme de pseudo-sérialisation est à réaliser en plusieurs étapes :

1. Stocker dans une chaine de caractères l'ensemble des propriétés de l'objet en séparant chacune d'elles par un (ou des) caractère(s) improbable(s).
2. Créer une méthode dans la classe de l'objet permettant de valoriser ses propriétés en fonction de la chaine issue de la sérialisation.



Comme en atteste les couleurs sur le schéma ci-dessus, les deux objets, bien qu'ils possèdent les mêmes propriétés, sont différents : la désérialisation a créé un nouvel objet disposant des mêmes propriétés que l'original.

Voici un exemple de classe clsPersonne :

```
Option Compare Database

'---------------Membres privés---------------
Private p_strNom As String
Private p_strPrenom As String
Private p_intAge As Integer

'-----------------Propriétés----------------
Public Property Get Nom() As String
    Nom = p_strNom
End Property
Public Property Let Nom(strNom As String)
    p_strNom = strNom
End Property
Public Property Get Prenom() As String
    Prenom = p_strPrenom
End Property
Public Property Let Prenom(strPrenom As String)
    p_strPrenom = strPrenom
End Property
Public Property Get Age() As Integer
    Age = p_intAge
End Property
Public Property Let Age(intAge As Integer)
    p_intAge = intAge
End Property

Public Sub SePresenter()
    MsgBox "Je m'appelle " & p_strNom & " " &
p_strPrenom & " et j'ai " & p_intAge & " ans"
End Sub
```

Nous allons la modifier pour qu'elle expose les méthodes permettant son clonage.

```
Public Function Serialiser() As String
    Serialiser = p_strNom & "#" & p_strPrenom & "#" &
p_intAge
End Function

Public Sub Deserialiser(strChaine As String)
    'Découpe de la chaine de sérialisation
    Dim strTemp() As String
    strTemp = Split(strChaine, "#")

    'Affectation aux membres privés
    p_strNom = strTemp(0)
    p_strPrenom = strTemp(1)
    p_intAge = CInt(strTemp(2))
End Sub
```

Exemple d'utilisation :

```
Sub Exemple()
Dim oPersonne1 As New clsPersonne
With oPersonne1
    .Nom = "DUPOND"
    .Prenom = "Paul"
    .Age = 22
End With

'Ajoute la variable temporaire
Application.TempVars.Add "Ma Personne",
oPersonne1.Serialiser
'Détruit l'objet oPersonne1
Set oPersonne1 = Nothing

'Recrée un objet identique à oPersonne1 depuis la
variable temporaire
Dim oPersonne2 As New clsPersonne
```

```
oPersonne2.Deserialiser Application.TempVars("Ma
Personne")

'Interroge le nouvel objet
oPersonne2.SePresenter

End Sub
```

7. Persistance des données

Bien qu'il s'agisse de données temporaires, il n'en reste pas moins que la question du stockage peut arriver à un moment ou un autre. En effet, si la durée de vie d'une donnée stockée dans une base se veut " éternelle ", celle d'une variable est limitée à celle de l'utilisation du projet. Le but de ce chapitre : proposer des alternatives permettant de stocker les informations ne faisant pas partie du domaine de gestion en tenant compte du fait que l'application soit ou non multi-utilisateurs.

Le domaine de gestion est un référentiel regroupant l'ensemble des données prises en compte lors de la modélisation de la base. Il peut s'agir d'un client, d'une commande, etc. Une variable d'application telle que nous l'avons vue plus haut dans cet article ne fait pas partie de ce domaine. Exemple : le nom de l'utilisateur de l'application.

7.1. Cas des applications mono-utilisateur et mono-poste

7.1.1. Problématique

Si vous utilisez le nouveau format de données accdb et les nouveautés du moteur ACE, il est fort probable que vous soyez dans cette situation, à savoir : un seul fichier Access déployé sur un seul poste.

Dans ce cas, le stockage des variables temporaires pour une utilisation ultérieure (autre session du système, après un arrêt, etc) est assez simple, il suffit de disposer d'une table où sera inscrite les variables à chaque modification.

La table nommée **tbl_util_variable** possèdera deux champs de type texte :

- **VarNom** : nom de la variable
- **VarValeur** : valeur de la variable

A l'ouverture de la base de données la macro **AutoExec** aura en charge de recréer chacune des variables définies dans la table. Ceci se fera notamment à l'aide d'un recordset DAO basé sur la table **tbl_util_variable**. La seule difficulté réside dans le choix de la technologie à utiliser pour écrire les variables dans la table. En effet, il n'existe pas d'évènement exécuté à la fermeture de l'application qui pourrait permettre de modifier le contenu de la table en fonction des éléments présents dans la collection **Application.TempVars**. Il sera donc nécessaire de modifier le contenu de la table à la volée (c'est-à-dire lors de la création, modification, ou suppression d'une variable temporaire). Malheureusement, une nouvelle fois, aucun système n'a été prévu dans ce sens et les variables temporaires ne déclenchent aucun évènement. Une solution consiste à créer sa propre classe encapsulant la collection **TempVars** de l'objet Application. Cette classe sera nommée **clsTempVars** et tous les appels **d'Application.TempsVars** seront remplacés par des instructions destinées à un objet de type **clsTempVars**.

La structure de base de la classe clsTempVars est la suivante :

```
Property Get TempVars() As TempVars
    Set TempVars = Application.TempVars
End Property

Property Let Item(strVarNom As String, strVarValeur As
String)
    Application.TempVars(strVarNom) = strVarValeur
End Property
Property Get Item(strVarNom As String) As String
    Item = Nz(Application.TempVars(strVarNom))
End Property

Sub Remove(strVarNom As String)
    Application.TempVars.Remove strVarNom
End Sub
Sub RemoveAll()
    Application.TempVars.RemoveAll
End Sub
Sub Add(strVarNom As String, strVarValeur As String)
    Me.Item(strVarNom) = strVarValeur
End Sub
```

Si avant l'affectation de la variable Essai à 100 était obtenu à l'aide de :

```
Application.TempVars("Essai") = 100
```

Il faut maintenant utilisé :

```
Dim oTmpVar As New clsTempVars
oTmpVar.Item("Essai") = 100
```

A première vue, l'intérêt est moindre. Cependant, les méthodes **Remove**, **Add**, et **Item** peuvent facilement être modifiées pour prendre en compte le traitement à appliquer aux enregistrements de la table **tbl_util_variable**. En comparaison avec des langages objets plus évolués (Java, DotNet), nous pourrions presque dire que la classe **clsTempVars** est un héritage de la classe TempVars et que ses différentes méthodes sont surchargées.

Bien entendu, du fait que les variables temporaires créées par cette classe sont ajoutées à la collection **Application.TempVars**, les instructions de consultations telles que celles ci-dessous sont toujours valides et les variables temporaires gardent une portée universelle (formulaire, requête, VBA, automation, etc.)

```
MsgBox Application.TempVars("Essai")
```

Pour créer la classe vue précédemment, vous devez créer un nouveau module de classe, l'enregistrer sous le nom clsTempVars et y écrire le code VBA donné.

7.1.2. Enrichissement de la classe clsTempVars pour la création des variables

Comme indiqué plus haut, au démarrage, la macro AutoExec sera chargée de consulter la table tbl_util_variable et de créer les différentes variables temporaires pour l'application. Un recordset DAO est nécessaire.

```
Sub Initialiser()

    'Nom de la table contenant les variables
    Const NOMTABLE = "tbl_util_variable"

    'Déclaration des variables DAO
    Dim oDb As DAO.Database
    Dim oRst As DAO.Recordset

    'Accès à la base de données
    Set oDb = CurrentDb
    'Ouverture du recordset
    Set oRst = oDb.OpenRecordset("tbl_util_variable",
dbOpenForwardOnly)

    With oRst
        'Parcours les enregistrements jusqu'à la fin du
recordset
        'et crée les variables temporaires une à une
        While Not .EOF
            Application.TempVars.Add .Fields(0).Value,
.Fields(1).Value
            .MoveNext
        Wend
        'Ferme le recordset
        .Close
    End With

    'Libère les ressources
    Set oRst = Nothing
    Set oDb = Nothing

End Sub
```

Voici un exemple de fonction lancé par la macro AutoExec :

```
Function AutoExec()

    'Crée les variables temporaires définies dans la
table tbl_util_variable
    Dim oTmpVar As New clsTempVars
    oTmpVar.Initialiser

End Function
```

7.1.3. Enrichissement de la classe clsTempVars pour l'écriture dans la table

Les méthodes **Item**, **Remove** et **RemoveAll** de la classe **clsTempsVars** doivent être modifiées afin que la table **tbl_util_variable** soit constamment la copie conforme de la classe **Application.TempVars**. La méthode **Add** n'a pas besoin de subir de modification puisqu'elle fait un simple appel à la méthode **Item**.

Les suppressions peuvent être obtenues à l'aide d'une requête.

```
Sub RemoveAll()
    'Déclaration DAO
    Dim oDb As DAO.Database

    'Vide la collection TempVars
    Application.TempVars.RemoveAll

    'Vide la table
    Set oDb = CurrentDb
    oDb.Execute "DELETE FROM tbl_util_variable",
dbFailOnError

    'Libère les ressources
    Set oDb = Nothing
End Sub

Sub Remove(strVarNom As String)
    'Déclaration DAO
    Dim oDb As DAO.Database

    'Supprime la variable temporaire
    Application.TempVars.Remove strVarNom

    'Vide la table
    Set oDb = CurrentDb
    oDb.Execute "DELETE FROM tbl_util_variable WHERE "
& BuildCriteria("VarNom", dbText, strVarNom),
dbFailOnError

    'Libère les ressources
    Set oDb = Nothing
End Sub
```

La propriété **Item** est un peu plus complexe puisqu'elle peut être utilisée pour ajouter une nouvelle variable ou modifier la valeur d'une existante. De ce fait, un **recordset** est nécessaire afin d'éviter l'insertion d'un doublon. (Une solution à base de requête **Update/Insert** combinée à une gestion d'erreur aurait aussi pu être utilisée dans ce cas de figure)

```
Property Let Item(strVarNom As String, strVarValeur As
String)

    'Décalaration des variables DAO
    Dim oDb As DAO.Database
    Dim oRst As DAO.Recordset

    'Modifie la variable temporaire
    Application.TempVars(strVarNom) = strVarValeur

    'Ouvre le recordset
    Set oDb = CurrentDb
    Set oRst = oDb.OpenRecordset("tbl_util_variable",
dbOpenDynaset)

    With oRst
        'Recherche la variable dans la table
        .FindFirst BuildCriteria("VarNom", dbText,
strVarNom)
        'Si non trouvé, alors crée l'enregistrement,
sinon modifie
        If .NoMatch Then
            .AddNew
            .Fields(0).Value = strVarNom
            .Fields(1).Value = strVarValeur
            .Update
        Else
            .Edit
            .Fields(1).Value = strVarValeur
            .Update
        End If
        'Ferme le recordset
        .Close
    End With

    'Libère les ressources
    Set oRst = Nothing
    Set oDb = Nothing
End Property
```

7.1.4. Expiration des données

Stocker les variables temporaires est intéressant. Toutefois il pourrait s'avérer très utile de limiter cette persistance dans la durée. A l'instar des cookies internet, la mise en place d'une date

d'expiration peut être envisagée. Dans un premier temps, il faut modifier la table tbl_util_variable afin d'y ajouter le champ VarExpire de type date qui définira la date au-delà de laquelle la donnée sera supprimée de la base. Ce champ pourra avoir la valeur NULL afin d'autoriser un stockage perpétuel.

La suppression des enregistrements dans la table peut être obtenue à l'aide d'une requête lancée depuis la classe clsTempVars.

```
Private Sub DetruireExpire()
    'Décalaration des variables DAO
    Dim oDb As DAO.Database

    'Accès à la base de données
    Set oDb = CurrentDb
    'Détruit les enregistrements expirés
    oDb.Execute "DELETE FROM tbl_util_variable WHERE
VarExpire<Now()", dbFailOnError

    'Libère les ressources
    Set oDb = Nothing
End Sub
```

Afin d'exécuter la procédure ci-dessus à chaque utilisation des variables temporaires, elle doit être invoquée dans le constructeur (Class_Initialize) de la classe.

```
Private Sub Class_Initialize()
    Call DetruireExpire
End Sub
```

La définition de la date d'expiration sera définie à l'aide de la méthode Expire.

```
Property Let Expire(strVarNom As String, dtVarExpire As
Variant)
    'Déclaration DAO
    Dim oDb As DAO.Database

    'Met à jour la date d'expiration
    Set oDb = CurrentDb
    oDb.Execute "UPDATE tbl_util_variable SET
VarExpire=" & _
                IIf(IsNull(dtVarExpire), "NULL", "#" &
Format(dtVarExpire, "mm/dd/yyyy hh:nn:ss") & "#") & _
                " WHERE " & BuildCriteria("VarNom",
dbText, strVarNom), dbFailOnError

    'Libère les ressources
    Set oDb = Nothing
End Property
```

Exemple d'utilisation :

```
Dim oTmpVar As New clsTempVars
oTmpVar.Item("Essai") = 100
oTmpVar.Expire("Essai") = Now() + 20
```

Retrouvez l'article complet de Christophe Warin en ligne : Lien84

Les derniers tutoriels et articles

Haute disponibilité avec MS SQL Server

La haute disponibilité est le fait de s'assurer des conditions optimales de fonctionnement en continu d'un système abritant un serveur SQL. Aujourd'hui ce genre d'exigence est au niveau des "5 neufs", c'est à dire une disponibilité du système de 99,999 %. Seules quelques grandes marques s'engagent sur de tels chiffres, comme HP, car avec un tel taux de disponibilité, vous n'avez droit qu'à un seul arrêt du système d'environ 5 minutes par an, soit juste le temps de passer un "Service Pack"

0. La haute disponibilité

La notion de haute disponibilité doit être établie par rapport à l'exigence de permance de la solution informatique.

La question fondamentale est :
quelles sont les données que l'on accepte de perdre en fonction du contexte de survenance d'un incident ?

La question sous jacente est :
Quel coût financier de la solution de continuité doit être envisagé en regard de la perte de production ?

Elle pose directement le problème d'un retour sur investissement, même si l'investissement à des chances de ne jamais être consommé (absence d'incident). C'est donc a un calcul de probabilité calqué sur les modèles des compagnies d'assurance qu'il faut se prêter.

0.1. La perte d'exploitation

La mesure de la perte se chiffre généralement en terme de durée d'exploitation perdue. A une durée d'exploitation perdue peut correspondre une durée de remise en état beaucoup plus longue afin de rétablir le système comme à l'origine.
Par exemple, à une perte d'exploitation des trois dernières minutes de production peut correspondre une durée de remise en état d'une heure.

0.2. Contexte de l'incident

Il convient de prendre en compte la force et la gravité de l'incident comme critère à intégrer dans le processus de haute disponibilité. Il n'est pas toujours possible, de prendre en compte tous les incidents sans tenir compte du coût ou de la logistique globale de mise en oeuvre de la solution de dépannage.
Par exemple en cas d'incendie ou doit considérer le redémarrage "global" du système : nouveaux locaux, information du personnel... autant de préalables qui finalement font du coût de la solution de haute disponibilité, un élément presque anecdotique.

On comprend donc qu'assurer une haute disponibilité avec une perte nulle de la production et pour tous les types d'incident est relève d'une gageure financière pas toujours en adéquation avec les budgets informatiques.

Ce document présente donc différentes approches pour ce faire en détaillant les avantages et les inconvénients de chacune des méthodes. Il ne s'occupe pas du volet "sécurité" qui relève du domaine des administrateurs de systèmes informatiques et des stratégies de l'entreprise.

1. Avant tout se prémunir de la défaillance

Se prémunir de la défaillance consiste à utiliser un serveur configuré de telle façon que les principaux incidents n'ait que peu ou pas d'influence sur la continuité de la production.

Dans ce cadre, il faut considérer au niveau matériel, un serveur configuré de la sorte :

- Une alimentation redondante
- Mémoire RAM autocorrective
- Disque RAID hot plug avec 3 agrégats (un RAID niveau 5 et deux RAID niveau 1)
- Onduleur online

On veillera de plus à stocker un disque hot plug en "spare" de façon à pallier immédiatement à la panne.

En outre il faut entretenir un serveur de secours capable de reprendre les mêmes disques hot plug que le serveur principal. Le serveur de secours étant configuré à l'identique au niveau hard et soft (OS + SQL Server).

Les fichiers seront répartis comme suit afin de réparer les défaillances :

- RAID 5 : fichiers contenant les données des bases (DATA)
- RAID 1 (grappe 2) : fichiers contenant les journaux des bases (LOG)
- RAID 1 (grappe 2) : autres fichiers (OS, exe, mémoire virtuelle...)

Pannes possibles	Remède
Coupure réseau éléctrique	Onduleur
Défaut de RAM	Auto correction soft
Défaut du sous système de contrôle disque (RAID, SCSI, SATA...)	Enficher * les disques DATA et LOG du serveur en panne dans le serveur de secours. Copier les fichiers nécessaire au redémarrage. Lancer les procédures adéquates pour reprendre la main sur SQL Server et la base de production.
Défaillance d'un disque	Remplacement à chaud du disque
Défaillance d'un fichier	Si fichier OS ou exe : enficher * les disques DATA et LOG du serveur en

	panne dans le serveur de secours. Copier les fichiers nécessaires au redémarrage. Lancer les procédures adéquates pour reprendre la main sur SQL Server et la base de production. Si fichier DATA SQL : procédure de reprise depuis fichier LOG SQL. Si fichier LOG SQL : procédure de reprise depuis fichier DATA SQL.
Défaillance processeur	Enficher * les disques DATA et LOG du serveur en panne dans le serveur de secours. Copier les fichiers nécessaire au redémarrage. Lancer les procédures adéquates pour reprendre la main sur SQL Server et la base de production.

* cette pratique n'est pas garantie par Microsoft. En effet, les fichiers des données comme ceux des journaux peuvent ne pas pouvoir être repris tel quel parce qu'il sont ouvert à l'usage exclusif de MS SQL Server tant que le serveur SQL tourne (en principe 24h/24).

Bien entendu il est toujours possible de repartir d'une sauvegarde mais la perte de production dans ce cas est lié au delta entre deux sauvegardes.

NOTA : MS SQL Server est capable de sauvegardes très légères (différentielles, JT) que l'on peut programmer à fréquences plus ou moins élevées (jusqu'au 1/4 d'heure par exemple).

2. La solution de clusterisation

C'est la seule solution actuellement garantie par MS en terme de reprise d'exploitation sans perte de données.

Pour cela il faut :

- Deux serveurs identiques, choisit dans une liste des matériels approuvés par Microsoft,
- Une baie de disque partagée, (SAN) choisit dans une liste des matériels approuvés par Microsoft,
- La solution MS Windows 2003 Server Clustering

Cette solution est couteuse en deux endroits :

- Le matériel approuvé par MS est très limité et ce sont des serveurs hauts de gammes (cette exigence disparait avec la version 2005 sous Windows Server 2003).
- L'installation du clustering nécessite des compétences étendue au niveau système.

Néanmoins il s'agit d'une solution simple en exploitation car elle est transparente et permet un basculement automatique sans même qu'aucune intervention humaine soit nécessaire.
le seul inconvénient technique de cette solution est un temps de latence du basculement qui est de quelques dizaines de secondes.

Article en français expliquant les principes du clustering : Lien85

Document en anglais de la solution MS : Lien86

Avantages	Inconvénients
Seule solutions garantie sans pertes de données. Pas ou peu d'administration Basculement automatique Faible coût d'exploitation Faible consomation des ressources du serveur	Couteuse en matériel et logiciel. Installation complexe nécessitant du personnel qualifié Concerne tout le serveur single point of failure" : le SAN...

Mais à bien y regarder, cette solution n'est pas toujours aussi couteuse qu'on le croit parce qu'un SAN économise du disque. En revanche, le SAN reste le talon d'achile car en cas de sérieux problème sur ce dispositif, c'est l'ensemble du système qui est en panne !

3. Les solutions logiques

3.1. Basculement avec sauvegardes

La solution de basculement avec sauvegarde consiste à mettre en exploitation sur un serveur de secours la plus récente sauvegarde de la base de données et repartir de cette sauvegarde pour commencer une nouvelle exploitation.

Les pertes engendrées dépendent donc :

- du temps de latence des sauvegardes pour la perte des données
- des points de reprise fonctionnels pour la perte d'exploitation (vague en cours ?)

Pour réduire la perte des données on peut agmenter les fréquences des sauvegardes. Dans ce cas un plan de sauvegarde alternant des sauvegardes complète la nuit, des différentielles en production et des sauvegardes du journal de transaction au fil de l'eau peuvent réduire considérablement les temps de latence et faire en sorte que la perte des données soit réduite à une moyenne de quelques minutes d'exploitation.

Avantages	Inconvénients
Très faible coût matériel (ne nécessite pas l'aquisition préalable d'un serveur de secours) Faible coût d'installation Faible consomation des ressources du serveur Concerne une ou plusieurs bases	Pertes importantes possibles Basculement manuel et long Nécessite une administration au quotidien

3.2. Log Shipping

Le log shipping (litteralement "envoi de journaux") est une technique qui consiste à envoyer de manière planifiée et régulière la partie du journal de la base qui contient les dernières transactions achevées.
En quelques sortes, le Log Shipping introduit de manière automatique la solution vue précédemment.
Les scripts de mise en oeuvre de cette technique figurent dans le CD de l'édition Entreprise de MS SQL Server et sa simplicité fait qu'elle peut être mise en oeuvre sur n'importe quel serveur de n'importe quelle édition.
Il est possible de descendre le temps de latence entre deux envois de journaux à quelques minutes, afin de minimiser les pertes.

Avantages	Inconvénients
Faible coût d'installation Très faible cout d'administration Faible consomation des ressources du serveur Concerne une ou plusieurs bases	Pertes moyennes possibles Basculement manuel

Dans la version SQL Server 2000, le Log Shipping constitue souvent la solution de meilleurs compromis entre le coût à tous

niveaux et la perte possible.

3.3. Réplication

La réplication consiste à dupliquer des informations d'une base à une autre. La granularité de la réplication permet de descendre jusqu'à la donnée unitaire (une colonne d'une ligne d'une table). Le système repose donc sur des articles qui sont des parties de tables (filtrage horizontal et vertical), qui sont mis à disposition sur un distributeur que les abonnés viennent reprendre.

Suivant les différents modèles de réplication, la reprise des données se fait par "push" ou "pull".

La réplication nécessite en fait 3 serveurs SQL : le répliqué (éditeur), le serveur de publication et le serveur abonné. Mais en fait le serveur de publication peut être l'un des deux serveurs, et dans le cas de solution de continuité, le mieux est que le serveur de publication soit aussi le serveur abonné.

Les différents modèle de réplications sont les suivants :

- capture instantanée
- transactionnelle
- fusion

Ces réplications peuvent être combinées avec une mise à jour immédiate de l'abonnée ou différée.

Dans le cas qui nous péoccupe (solution de continuité) il convient de choisir la solution de réplication dont le temps de latence peut être le plus court. Dans ce cadre il convient donc de préférer la réplication transactionnelle avec mise à jour immédiate. Le gros inconvénient dans ce cas est que les modifications du schéma ne sont pas prises en compte et nécessite un script à jouer simultanément sur les deux serveurs ainsi qu'une modification de la définition des objets répliqués.

Avantages	Inconvénients
Ne nécessite pas de basculement Concerne une ou plusieurs bases, voire même un élément (sous ensemble d'une table)	Cout d'installation élevé Consomation élevée des ressources du serveur suivant la position du distributeur Cout d'exploitation élevé Pertes faibles à moyenne possibles Les modifications de schéma peuvent ne pas être prise en compte suivant le mode de réplication

Cette solution est celle qui minimise les pertes après le clustering, mais son coût, lié à sa complexité est très élevé, en particulier dans les bases de données dont la structure est sujette à modification.

3.4. Mirroring

Le mirroring est un concept nouveau introduit avec la version 2005 de MS SQL Server. Ce technique de mise en mirroir est simple et pratique. Elle consiste avant tout à dupliquer intégralement une base de données en temps réel, base qui sera alors accessible en lecture seule tant que le mirroir est actif et en lecture écriture si le mirroir est brisé. Comme dans le cas du clustering, la reprise peut être automatisé et en pratique le basculement s'opère en quelques dizaines de secondes, à condition d'avoir mis en place un serveur de scrutation, qui peut être constitué par une machine base de game (simple PC) avec le run time SQL Server (Express 2005).
Cette solution n'exige en outre aucun matériel spécifique.

En terme de richesse, souplesse et coût, cette solution s'avère la mieux adaptée pour la haute disponibilité d'un faible nombre de bases de données (quelques unités).

Avantages	Inconvénients
Pas de surcoût matériel Basculement automatique possible Concerne une ou plusieurs bases Pas de pertes de données Pas ou peu d'administration Faible coût d'installation Faible coût d'exploitation Faible consomation des ressources du serveur	Disponible uniquement en version 2005

S'il y avait une solution à retenir dans le cadre d'epsilon, ce serait certainement celle-là qui possède tous les avantages mais nécessite la validation de notre logiciel dans la version MS SQL Server 2005.

Retrouvez l'article de Frédéric Brouard en ligne : Lien87

Gestion des transactions imbriquées

Une grosse difficulté qui attendent les développeurs est de savoir comment piloter les transactions dès lors que celle-ci s'emboitent les unes dans les autres notamment lors des appels de procédures stockées. C'est ce que l'on appelle les transactions imbriquées.
Cet article présente sommairement la difficulté et le moyen de gérer le plus proprement possible de telles transactions dans le cadre d'un développement recourant généreusement aux procédures stockées.

1. Ce qu'est..., ce que n'est pas... une transaction

Une transaction est un ensemble de traitements devant être effectué en tout ou rien, en vertue du principe d'atomicité des transaction. Par exemple un virement bancaire d'un compte courant à un compte éparge nécessite une première requête UPDATE pour soutirer l'argent du compte courant et une seconde pour créditer le compte épargne. Si l'une des deux requêtes ne s'effectue pas, alors la base devient incohérente. On dit ainsi que la transaction assure que la base de données part d'un état de cohérence pour arriver dans un autre état de cohérence, les états transitoires, c'est à dire les différentes étapes de la transaction, ne devant jamais être présentés de quelques manière que ce soit, même en cas de panne du système.

Mais que se passe t-il si une transaction démarre à l'intérieur d'une autre transaction ? C'est ce que l'on apelle "tansaction imbriquée". Les transactions imbriquées sont le plus souvent le fait de procédures stockées qui s'appellent les unes des autres afin de fournir un ensemble cohérent de traitement donc chaque partie peut en outre être individuellement appelée.

Le cas est assez classique. On le trouve par exemple lorsque le modèle de données cartographie un objet et que différentes procédures concourent à l'insertion de ses données comme à sa mise à jour. Par exemple une première procédure gère la mise à jour (INSERT / UPDATE / DELETE) d'une personne et appelle une seconde procédure qui gère la mise à jour des adresses relatives à cette personne. D'où deux transactions (une dans chaque procédure) qui fatalement vont s'imbriquées.

Par exemple une procédure stockée 1 démarre une transaction et au milieu de code, alors que la transaction 1 n'est pas finalisée, fait appel à une autre procédure stockée qui elle même encapsule une procédure stockée... Qui valide finalement la transaction ? La procédure appelante ou celle qui est appelée ? Qui annule finalement la transaction ?

Or le principe même d'une transaction imbriquée n'a pas de sens. En effet une transaction est un ensemble cohérent. Imaginons le scénario suivant :

```
BEGIN TRANSACTION A

... code a1 ...

   BEGIN TRANSACTION B

   ... code b ...

   ROLLBACK TRANSACTION B

... code a2 ...

COMMIT TRANSACTION A
```

La transaction B valide les parties de code a1 et a2, mais le code b étant annulé, la transaction A est clairement incohérente. C'est pourquoi dans le principe les transactions imbriquées ne sont pas possible !
En fait il n'y a donc jamais qu'une seule transaction. Et c'est toujours la première...

2. Modèle de transaction imbriquées

Dès lors deux modèles de "pseudo" transactions imbriquées sont possibles : le modèle symétrique et le modèle asymétrique.

Dans le modèle symétrique, le premier BEGIN TRANSACTION commence la vraie seule transaction. Chaque fois qu'un nouveau BEGIN TRANSACTION est rencontré dans le code, la commande est ignorée, mais un compteur de transaction est incrémenté de 1. Chaque fois qu'un COMMIT ou ROLLBACK est rencontré, ce même compteur est décrémenté de 1 et la commande n'a pas d'effet. Si le compteur est à zéro, alors le COMMIT ou ROLLBACK rencontré est réellement exécuté. Cela peut se résumé par le script suivant :

```
BEGIN TRANSACTION A

... code a1 ...

   BEGIN TRANSACTION B -- code ignoré, compteur "tran"
à 1

   ... code b ...

   ROLLBACK TRANSACTION B -- code ignoré, compteur
"tran" à 0

... code a2 ...

COMMIT TRANSACTION A
```

Ainsi comme on le voit, tout le code de cette procédure est exécuté.

Mais ce n'est pas le comportement adopté par MS SQL Server... En effet, ce SGBDR se base sur le modèle asymétrique, finalement bien plus fin !

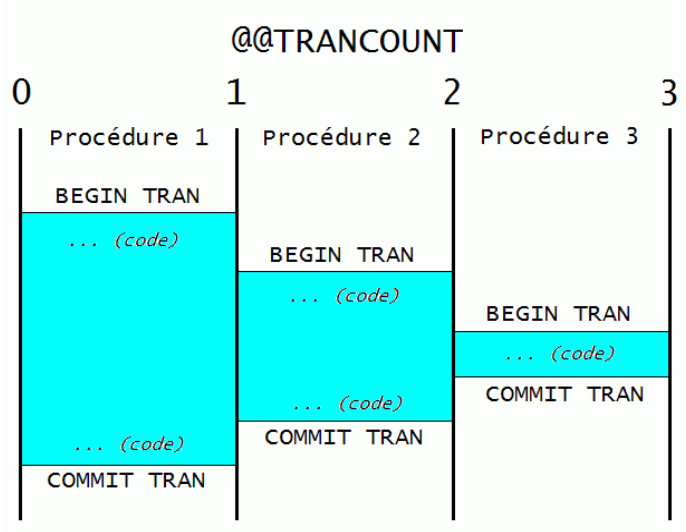
3. Le modèle asymétrique de transaction imbriqué

Le principe du modèle asymétrique de transactions imbriquées est simple, mais sa mise en œuvre réserve quelques surprises !

Voici les règles de base :

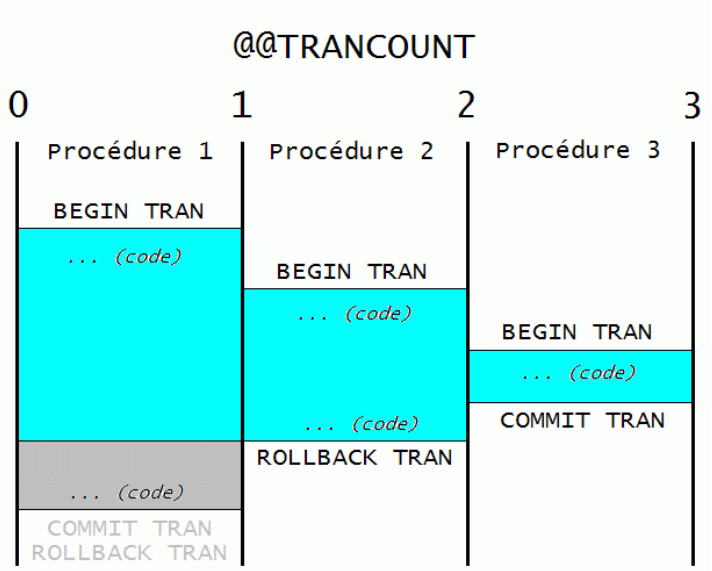
1. il n'y a jamais qu'une seule transaction
2. Le premier BEGIN TRANSACTION rencontré démarre la transaction
3. tout autre BEGIN TRANSACTION que le premier ne fait qu'incrémenter le compteur de session @@TRANCOUNT
4. le premier ROLLBACK TRANSACTION rencontré annule la transaction
5. chaque COMMIT TRANSACTION décrémente le compteur de session @@TRANCOUNT de 1 et si ce compteur vaut 0 alors la transaction est finalement validée.

Voici ce qui se passe lorsque des transactions imbriquées réussissent :



Dans cet exemple, la procédure 1 démarre une transaction avec un BEGIN TRANSACTION et met le compteur @@TRANCOUNT à 1, puis appelle la procédure 2 qui, voyant qu'une transaction est déjà démarrée, ne fait que mettre le compteur @@TRANCOUT à 2, puis appelle la procédure 3 qui, voyant qu'une transaction est déjà démarrée, ne fait que mettre le compteur @@TRANCOUT à 3. Cette dernière transaction réussie et ne fait que décrémenter le compteur @@RANCOUNT qui passe de 3 à 2 puis revient en procédure 2, qui elle même réussit aussi et ne fait que passer le compteur @@TRANCOUNT de 2 à 1. Enfin la procédure 1 fait le commit final qui fait passer @@TRANCOUNT de 1 à 0 et génère réellement le COMMIT !
En tout et pour tout il n'y a eût qu'un seul BEGIN TRANSACTION et un seul COMMIT TRANSACTION.
La notion même de transaction imbriquée n'existe donc pas...

Que se passe t-il si une transaction interne génère un rollback ?



En fait dans ce cas le ROLLBACK est immédiatement exécuté et le compteur @@TRANCOUNT passe à zéro.
Si jamais le code dans procédure 1 passe par le COMMIT, alors le système ne s'y retrouve plus et génère un message d'erreur du genre : Le compte des transactions après EXECUTE indique qu'il manque une instruction COMMIT ou ROLLBACK TRANSACTION

Démonstration :

```
-- création d'une table test pour notre transaction
IF EXISTS (SELECT *
           FROM    INFORMATION_SCHEMA.TABLES
           WHERE   TABLE_SCHEMA = 'dbo'
             AND   TABLE_NAME = 'T_TRN')
   DROP TABLE T_TRN
GO

-- table avec une contrainte de validité
CREATE TABLE T_TRN
(N INT CHECK (N >= 0))
GO

-- création d'une procédure stockée de test de
transaction imbriquée
IF EXISTS (SELECT *
           FROM    INFORMATION_SCHEMA.ROUTINES
           WHERE   ROUTINE_SCHEMA = 'dbo'
             AND   ROUTINE_NAME = 'P_TRN_INTERNE')
   DROP PROCEDURE P_TRN_INTERNE
GO

CREATE PROCEDURE P_TRN_INTERNE
AS

DECLARE @ERROR INT, @ROWCOUNT INT

BEGIN TRANSACTION

-- insertion invalide : elle doit déclencher le
ROLLBACK
INSERT INTO T_TRN VALUES (-4)
SELECT @ERROR = @@ERROR, @ROWCOUNT = @@ROWCOUNT
IF @ERROR <> 0 OR @ROWCOUNT = 0
BEGIN
   RAISERROR('Procédure P_TRN_INTERNE : Erreur à
l''insertion', 16, 1)
   GOTO LBL_ERROR
END

COMMIT TRANSACTION

RETURN (0)

LBL_ERROR:

IF @@TRANCOUNT > 1
   COMMIT TRANSACTION
IF @@ROWCOUNT = 1
   ROLLBACK TRANSACTION
RETURN (-1)

GO

-- création d'une procédure stockée de test de
transaction imbriquée
IF EXISTS (SELECT *
           FROM    INFORMATION_SCHEMA.ROUTINES
           WHERE   ROUTINE_SCHEMA = 'dbo'
             AND   ROUTINE_NAME = 'P_TRN_EXTERNE')
   DROP PROCEDURE P_TRN_EXTERNE
GO

CREATE PROCEDURE P_TRN_EXTERNE
AS

DECLARE @ERROR INT, @ROWCOUNT INT, @RETVAL INT

BEGIN TRANSACTION

-- insertion valide
INSERT INTO T_TRN VALUES (33)
SELECT @ERROR = @@ERROR, @ROWCOUNT = @@ROWCOUNT
```

```
IF @ERROR <> 0 OR @ROWCOUNT = 0
BEGIN
   RAISERROR('Procédure P_TRN_EXTERNE : Erreur à
l''insertion', 16, 1)
   GOTO LBL_ERROR
END

EXEC @RETVAL = P_TRN_INTERNE
SELECT @ERROR = @@ERROR, @ROWCOUNT = @@ROWCOUNT

IF @RETVAL = -1 -- la transaction a été pseudo validée
mais elle doit être annulée
BEGIN
   RAISERROR('Procédure P_TRN_EXTERNE : Erreur à
l''appel de la procédure P_TRN_INTERNE', 16, 1)
   GOTO LBL_ERROR
END

IF @ERROR <> 0 OR @@ROWCOUNT = 0
   GOTO LBL_ERROR

COMMIT TRANSACTION

RETURN (0)

LBL_ERROR:

IF @@TRANCOUNT > 1
   COMMIT TRANSACTION
IF @@ROWCOUNT = 1
   ROLLBACK TRANSACTION
RETURN (-1)

GO
-- exécution teste
EXEC P_TRN_EXTERNE
GO

-- a l'issu de cet exécution aucune ligne ne doit avoir
été inséré :
SELECT * FROM T_TRAN
GO
```

4. Piloter génériquement des transactions imbriquées

Si vous voulez piloter proprement des transactions qui s'emboitent dans d'autres transactions notamment lorsque vous faîtes appel à des procédures stockées qui s'imbriques les unes dans les autres il faut gérer le COMMIT ou le ROLLBACK en tenant compte de la valeur du compteur @@TRANCOUNT.

Voici comment finaliser proprement une transaction quelque soit le contexte transactionnel :

```
-- partie à rajouter à TOUTES les procédures
(finalisation) :

-- succès
COMMIT TRANSACTION
RETURN (0)

-- échec
LBL_ERROR:
IF @@TRANCOUNT > 1
   COMMIT TRANSACTION
IF @@ROWCOUNT = 1
   ROLLBACK TRANSACTION
RETURN (-1)
```

Si vous avez besoin de rétablir le niveau d'isolation par défaut :

```
...
DECLARE @RETVAL INT
...

-- succès
COMMIT TRANSACTION
SET @RETVAL = 0
GOTO RESUME

-- échec
LBL_ERROR:
IF @@TRANCOUNT > 1
   COMMIT TRANSACTION
IF @@ROWCOUNT = 1
   ROLLBACK TRANSACTION
SET @RETVAL = -1

LBL_RESUME:
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
```

Retrouvez l'article de Frédéric Brouard en ligne : Lien88

Packet Filter sur les systèmes BSD : les tables

Que sont les tables ? Pourquoi et comment les utiliser ?
Celles-ci sont utilisées par certains programmes externes pour interagir avec Packet Filter afin de traiter différemment de nouvelles adresses sans avoir à relire les règles. Vous vous rendrez vite compte de leur intérêt en les mettant à profit par exemple pour la constitution de listes noires (ou blanches) dynamiques.

1. Présentation

1.1. Une amélioration notable des listes et macros

Une table permet le regroupement d'adresses au sein d'une même entité. En quoi sont-elles différentes aux listes que l'on peut définir avec Packet Filter et que l'on peut réutiliser grâce aux macros (voir exemple ci-dessous) ?

```
##### Macros #####
me = "sis0"
tcpflags = "flags S/SFRA"
workstations = "{ 192.168.0.1 192.168.0.2 192.168.0.3
192.168.0.4 192.168.0.5 }"

##### Règles #####
block all
pass in quick inet proto tcp from $workstations to $me
port ssh $tcpflags keep state
pass in quick inet proto tcp from $workstations to $me
port http $tcpflags keep state
```

Les spécificités des tables par rapport aux listes standard sont les suivantes :

- Utilisation directe et multiple dans les règles, comparable à une macro
- Les recherches d'une adresse sont beaucoup plus rapides (utilise moins de mémoire et de temps CPU qu'une simple liste)
- Existence qui va au-delà du fichier de configuration où elles sont définies puisqu'elles résident ensuite en mémoire ce qui permet de leur appliquer des modifications au niveau de leur contenu. Ces modifications sont immédiatement prises en compte par Packet Filter sans avoir à recharger les règles ou à procéder à une quelconque manipulation.

1.2. Syntaxe

1.2.1. Déclaration d'une table

Le contenu d'une table est sensiblement identique à celui que l'on peut fournir comme adresse source ou de destination dans les règles en temps normal. Il peut prendre les différentes formes suivantes :

- Une adresse IP (version 4 comme 6), exemples : 192.168.0.1, 66.80.97.134
- Une adresse en notation CIDR (version 4 comme 6) : 192.168.0.0/24
- Un nom de machine, comme www.developpez.com, qui sera automatiquement remplacé par les adresses IP correspondantes (versions 4 et 6) à condition de pouvoir effectuer cette résolution (ne pas oublier de construire des règles visant à autoriser le trafic DNS)
- Le nom d'une interface (sis1, lo0, etc), qui sera substituée par la liste des adresses qui lui sont attribuées
- Le nom d'une interface suivie de la notation CIDR, exemple sis0/8
- Le nom d'une interface suivie de :network, :broadcast, :peer ou :0 correspondant respectivement à l'adresse réseau, l'adresse de broadcast, l'adresse d'un pair sur un lien point à point ou l'adresse principale affectée à une carte réseau (ne tient pas compte des alias)
- Le mot clé self
- L'une des formes antérieures précédée d'une négation

Exemples de déclarations de quelques tables :

```
table <sites_http_autorises> { www.developpez.net,
www.developpez.com, www.developpez.be }
table <serveurs_dns_internes> { 192.168.0.1,
192.168.0.101 }
table <blacklist> { 125.60.86.214 208.159.106.199
81.237.54.31 76.89.10.213 }
table <reseaux_internes> { sis1:network sis2:network }
```

Les signes < et > ne sont pas le résultat d'une erreur typographique et encadrent systématiquement le nom d'une table. On peut ainsi reconnaître d'un simple coup d'oeil qu'il s'agit d'une table, tout comme les macros sont précédées du signe dollar ($).

Les virgules séparant les différents éléments de la liste sont facultatives. Libre à vous donc de les omettre ou de les faire figurer.

L'utilisation des noms de machine (DNS) est à éviter autant que possible pour des raisons de sécurité.

L'utilisation d'adresses dynamiques (protocole DHCP) n'est ici pas gérée puisque les éléments d'une table sont systématiquement traduits en une adresse IP à leur intégration dans la table.

1.2.2. Utilisation dans les règles

Les tables peuvent être employées dans les différentes règles de manière quasi similaire à une adresse. Voici une liste exhaustive des emplacements où elles peuvent figurer :

- Comme ensemble d'adresses source ou de destination dans les règles de filtrage (pass et block)
- Comme ensemble d'adresses source ou de destination dans les règles de traduction d'adresse (nat et binat)
- Comme ensemble d'adresses source ou de destination dans les règles de redirection (rdr)

- Pour constituer une liste d'attaquants à l'aide du mot-clé overload suivant le comportement de l'hôte distant (la mesure est effectuée en nombre de connexions sur un certain laps de temps)

Lors de l'emploi d'une table dans une règle, il suffit de reprendre son nom à l'endroit souhaité sans oublier de l'encadrer de ses signes distinctifs < et >. Petit exemple pour illustrer :

```
##### Macros #####
if = "sis0"
tcpflags = "flags S/SFRA"

##### Tables #####
# Liste noire des machines dont l'activité est suspecte
table <blacklist> { 125.60.86.214 208.159.106.199
81.237.54.31 76.89.10.213 }
# Liste des serveurs web recencés
table <serveurs_http> { 192.168.0.102, 192.168.0.103 }

##### Règles #####
# Les paquets des machines blacklistées sont bloqués
nets
block in quick on $if from <blacklist> to any
# Le trafic vers les serveurs Web est autorisé (ainsi
que leurs réponses)
pass in quick on $if inet proto tcp from any \
       to <serveurs_http> port { http https } $tcpflags
keep state
```

Les tables ne font l'objet d'aucune restriction au niveau de leur emplacement. Elles doivent cependant être "déclarées" avant une quelconque référence dans vos règles.

2. Manipulation en ligne de commande avec pfctl

La commande pfctl accepte en paramètre exactement les mêmes types d'informations comme adresse (adresses IP, noms DNS, interfaces, ...) que cité précédemment. Ajoutons que celle-ci n'est d'ailleurs pas cantonnée à l'unique manipulation des tables.

Les exemples qui suivent seront basés sur la déclaration d'une table nommée blacklist visant à interdire les paquets de certaines adresses IP suite à des activités plus ou moins obscures :

```
table <blacklist> { 125.60.86.214 208.159.106.199
81.237.54.31 76.89.10.213 }
```

2.1. Limitations de pfctl

Il faut savoir que pfctl est soumis à certaines restrictions de la part du système et de l'implémentation de la commande elle-même :

- La commande pfctl n'altère que le contenu des tables en mémoire et ne répercute en aucun cas les différents changements demandés sur un quelconque fichier de configuration. Si vous avez besoin de synchroniser ces modifications en vue de les réappliquer au prochain démarrage de Packet Filter vous devrez le faire manuellement ou éventuellement à l'aide d'un script.
- Il vous sera impossible d'agir sur les tables à l'aide de pfctl si le niveau de sécurité du noyau atteint la valeur 2 sur NetBSD ou OpenBSD et la valeur 3 pour FreeBSD. Cette restriction concerne de manière plus générale les règles de filtrage qui sont ainsi chargées une bonne fois pour toutes avant le changement de ce niveau de sécurité.

2.2. Lister les adresses d'une table

Vous pouvez à tout moment consulter le contenu d'une table à l'aide de pfctl. Illustration avec notre table blacklist :

pfctl (-v) -t blacklist -T show

Qui produira la sortie suivante :

```
   76.89.10.213
   81.237.54.31
   125.60.86.214
   208.159.106.199
```

2.3. Ajouter des adresses à une table

Imaginons maintenant que nous voulions ajouter de nouvelles adresses à notre liste noire :

```
pfctl (-v) -t blacklist -T add 216.194.109.203
88.240.211.27
```

Notre table sera désormais composée des six adresses suivantes :

```
   76.89.10.213
   81.237.54.31
   88.240.211.27
   125.60.86.214
   208.159.106.199
   216.194.109.203
```

La table sera créée si celle-ci n'existe pas déjà.

2.4. Supprimer des adresses à une table

Nous avons besoin de supprimer deux adresses de celle-ci :

```
pfctl (-v) -t blacklist -T delete 125.60.86.214
76.89.10.213
```

Cette table contiendra de nouveau quatre adresses :

```
   81.237.54.31
   88.240.211.27
   208.159.106.199
   216.194.109.203
```

2.5. Recharger le contenu de la table

Nous souhaitons réinitialiser une table à partir de sa déclaration :

```
pfctl (-v) -t blacklist -T load -f /root/pf.conf
```

L'option -f est nécessaire pour indiquer le fichier où est définie la table en question.

Nous obtenons bien le contenu tel que la table a été déclarée :

```
   76.89.10.213
   81.237.54.31
   125.60.86.214
   208.159.106.199
```

2.6. Remplacer l'ensemble des adresses

Il est possible d'aller un peu plus loin en remplaçant l'intégralité des adresses actuelles d'une liste par de nouvelles. Pour cela on utilise la sous-commande replace, exemple :

```
pfctl (-v) -t blacklist -T replace 115.79.79.48
143.59.174.112 139.102.183.50
```

Le contenu sera désormais le suivant :

```
   115.79.79.48
```

```
139.102.183.50
143.59.174.112
```

La sous-commande replace, comme add, crée la table si cette dernière n'existe pas.

2.7. Tester si une adresse fait partie de la table

pfctl vous permet même de savoir si une adresse est actuellement définie dans la table donnée :

```
pfctl (-vv) -t blacklist -T test 115.79.79.48
www.developpez.com
```

Indiquera qu'il y a correspondance avec l'une des deux adresses (sur 115.79.79.48 et aucune avec www.developpez.com).

2.8. Les statistiques d'une table

Vous pouvez obtenir différentes informations (nombre de paquets bloqués ou passés) sur une table en particulier avec la commande suivante :

```
pfctl -vv -t blacklist -T show
```

Ou bien sur l'ensemble des tables avec :

```
pfctl -vv -s Tables
```

Pour réinitialiser les statistiques d'une table vous pouvez utiliser la sous-commande zero. Exemple avec notre table blacklist :

```
pfctl -t blacklist -T zero
```

2.9. Vider/Supprimer une table

Vous pouvez bien évidemment vider la table :

```
pfctl -t blacklist -T flush
```

Ou encore la supprimer mais celle-ci n'aura plus aucune existence en mémoire et ne sera plus manipulable via pfctl :

```
pfctl -t blacklist -T kill
```

Les règles faisant référence à une table supprimée subsistent mais deviennent donc, par conséquent, inutiles. Vous pouvez la recréer à tout moment à l'aide des commandes add ou replace (voir plus haut).

3. Les différentes propriétés apposables aux tables

3.1. Les tables immuables

Il est possible de rendre une table insensible à toute modification. Cet état est introduit par le mot-clé const. En effet, par défaut (en son absence) une table peut être modifiée par l'utilitaire pfctl, que nous avons étudié en détail ci-dessus.

Voici un exemple d'une telle table qu'il ne nous sera pas possible d'altérer même pour y ajouter une adresse :

```
table <serveurs_dns> const { 192.168.0.1
192.168.0.101 }
```

3.2. Les tables persistantes

Par défaut, le noyau se débarrasse automatiquement des tables inutilisées dans le sens où elles ne sont pas référencées par une quelconque règle. Cela peut s'avérer particulièrement gênant lorsque cette table est amenée à être utilisée par un programme externe et dont des règles sont ajoutées ou supprimées en fonction des besoins, bien souvent à l'aide du mécanisme appelé ancre (exemples courants : authpf ou encore certains mandataires FTP intégrés à pf).

La solution consiste à utiliser le mot-clé persist, rendant ainsi la table résidente en mémoire quoiqu'il advienne. Exemple de déclaration d'une table vide dont la persistance est assurée :
table <blacklist> persist

Cette table, blacklist, ici vide vous sera donc toujours accessible avec pfctl. Notez que vous pouvez également initialiser cette table à l'aide d'une ou plusieurs valeurs :

```
table <blacklist> persist { 67.80.122.54 67.80.122.55 }
```

3.3. Les tables dont le contenu réside dans un fichier

Packet Filter vous offre également la possibilité de stocker les adresses de votre table dans un fichier annexe, introduisant de ce fait un nouveau mot-clé : file. Voyons tout d'abord comment déclarer ce genre de tables quelque peu particulier à l'aide d'un exemple :

```
table <whitelist> file "/root/whitelist"
```

Puis on place les adresses, pouvant prendre l'une des formes citées lors de la présentation, à raison d'une adresse par ligne, dans le fichier indiqué. Ce dernier devra impérativement exister même vide sous peine de voir apparaître des erreurs fatales lors du chargement des règles. A noter, qu'il est même possible d'y insérer des commentaires, en les faisant précéder d'un caractère dièse (#). Voici le contenu de la table whitelist dont on prendra soin de mettre dans le fichier /root/whitelist tel que convenu :

/root/whitelist :

```
# Developpez
www.developpez.com
www.developpez.net
www.developpez.be
# Partenaires
43.204.225.199
75.186.215.37
```

Quand l'utilisation de ce type de table est-il justifié ? Lorsque le nombre d'adresses devient conséquent ou bien lorsque vous avez besoin d'ajouter ou de supprimer des adresses et de mémoriser ce changement d'état pour le prochain démarrage de Packet Filter, ce format étant fortement pratique pour des manipulations à l'aide de commandes shell par exemple. Ainsi pour ajouter une nouvelle adresse on pourrait utiliser un script semblable à celui-ci :

```
#!/bin/sh

# Partie concernant les paramètres du programme
progname=`basename $0`
args=`getopt t:f: $*`

usage() {
    echo "Usage: $progname -t table -f fichier adresse1
... adresseN"
    exit 2
}

[ $? -eq 0 ] || usage
```

```
set -- $args

for i
do
    case "$i"
    in
        -t)
            table="$2"
            shift; shift;;
        -f)
            fichier="$2"
            shift; shift;;
        --)
            shift; break;;
    esac
done

[ -z "$fichier" -o -z "$table" -o $# -lt 1 ] && usage

# On ajoute les adresses à la table et au fichier
correspondant
pfctl -t $table -T add $*
[ $? -eq 0 ] || exit $?
for arg in $*
do
    echo "$arg" >> $fichier
done
```

3.4. Combinaison de plusieurs propriétés

Il est parfaitement possible de combiner plusieurs voir les trois états (persist, const et file) abordés dans cette partie à la fois. Ainsi, l'exemple qui présente le plus d'intérêt serait de réaliser une table dont le contenu est chargé à partir d'un fichier annexe et déclarée constante pour qu'elle ne puisse pas être modifiée par l'utilitaire pfctl :

```
# En plaçant const avant file
table <blackist> const file "/root/blacklist"
# En plaçant const après file
table <blacklist> file "/root/blacklist" const
```

4. Autres éléments syntaxiques liés aux tables

4.1. Le mot-clé "self"

Ce mot-clé particulier est substitué par l'ensemble des adresses IP utilisées par votre machine. Ceci inclut également votre interface réseau de bouclage (également appelée loopback sous l'identifiant lo0 ou lo). Exemple, soit la configuration réseau suivante :

```
[root@freebsd ~]# ifconfig
de0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST>
mtu 1500
        inet 192.168.0.1 netmask 0xffffff00 broadcast
192.168.0.255
        ether 00:03:ff:3d:94:f9
        media: Ethernet autoselect (100baseTX)
        status: active
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu
16384
        inet6 ::1 prefixlen 128
        inet 127.0.0.1 netmask 0xff000000
```

self retournera les adresses suivantes :

- l'adresse IP attribuée à la carte réseau de0 : 192.168.0.1
- les adresses IP respectivement en version 4 et 6 affectées à la boucle locale : 127.0.0.1 et ::1

4.2. Les négations

Les négations sont utilisées pour exclure un sous-ensemble (adresse ou sous-réseau) d'un ensemble plus important (réseau) qui peut éventuellement figurer dans la même table. Afin d'exclure une adresse ou un ensemble (notation CIDR), il suffit de faire précéder cet élément d'un point d'exclamation (!). Prenons un exemple pour y voir plus clair :

On dispose d'un important réseau interne privé (rfc1918) découpé en plusieurs sous-réseaux dont un exclusivement destiné au service comptabilité dont on souhaite interdire l'accès à Internet, en ayant recours à une table, à l'exception d'un serveur situé dans ce service. Les affectations sont les suivantes :

- 192.168.0.0/16 : le réseau interne dans son intégralité
- 192.168.2.0/24 : le réseau interne réservé à la comptabilité
- 192.168.2.1 : un serveur (dans la partie comptabilité) effectuant des transferts programmés avec l'extérieur

La table qui découle de cette politique est alors la suivante :

```
table <internet_autorise> { 192.168.0.0/16 !
192.168.2.0/24 192.168.2.1 }
```

Et les règles semblables à celles-ci :

```
##### Macros #####
# Interface interne
int_if = "sis0"
tcpflags = "flags S/SFRA"

# Bloquer par défaut
block all
# Autorise les paquets TCP sortants vers Internet,
ainsi que leurs réponses
pass in quick on $int_if inet proto tcp from
<internet_autorise> \
       to any port { http https ... } $tcpflags keep
state
```

La correspondance se fera sur l'adresse la plus spécifique. Examinons quelques cas :

- 192.168.3.63 : la correspondance sera établie sur l'entrée 192.168.0.0/16 de la table, le paquet sera autorisé à passer
- 192.168.2.47 : aucune correspondance car toute la tranche 192.168.2.X est exclue à l'aide de la négation, le paquet sera bloqué
- 192.168.2.1 : la correspondance aura lieu sur 192.168.2.1 - la négation du sous-réseau 192.168.2 sera ignoré car son poids est moins important que celui de l'adresse elle-même, le paquet passera

Vous pouvez procéder à ces tests en utilisant la commande présentée dans Tester si une adresse fait partie de la table en faisant apparaître l'option de verbosité maximale (-vv).

4.3. Les listes

Ces mêmes listes qui nous ont permis d'introduire les tables peuvent parfaitement contenir une ou plusieurs tables comme éléments. Encore mieux elles peuvent contenir à la fois des tables et des adresses. En voici un exemple :

```
ext_if = "de0"
maison = "83.241.141.246"

table <partenaires> const { 73.184.228.146 24.1.3.149
```

```
186.220.104.61 88.240.211.27 }
table <collegues> { 142.80.125.240 131.254.222.237 }

# [...]

pass in quick on $ext_if from { <partenaires> $maison
<collegues> } to $ext_if

# [...]
```

La règle pass pourrait très bien être écrite sous des formes moins condensées équivalentes :

```
pass in quick on $ext_if from { <partenaires>
<collegues> } to $ext_if
pass in quick on $ext_if from $maison to $ext_if

# ou encore

pass in quick on $ext_if from <partenaires> to $ext_if
pass in quick on $ext_if from <collegues> to $ext_if
pass in quick on $ext_if from $maison to $ext_if
```

5. Utilitaires autour des tables

5.1. expiretable : supprimer des entrées obsolètes à intervalle régulier

Ce petit programme permet de retirer toutes les entrées qui ont passées plus d'un certain temps, que vous aurez fixé, au sein d'une table.
Son exécution requiert les droits de lecture et d'écriture sur le fichier spécial /dev/pf.

Installation :

FreeBSD, il ne nous sera pas nécessaire d'aller bien loin puisque vous pouvez trouver expiretable parmi les logiciels portés :

```
cd /usr/ports/security/expiretable
make install
```

Il en est de même pour **OpenBSD** :

```
cd /usr/ports/sysutils/expiretable
make install
```

En revanche il en est tout autrement sur **NetBSD** puisque ce logiciel ne figure pas parmi les paquetages :

```
cd ~
ftp http://expiretable.fnord.se/expiretable-0.6.tar.gz
tar xzf expiretable-0.6.tar.gz
ftp http://julp.developpez.com/bsd/packet-
filter/tables/fichiers/netbsd-expiretable-0.6.patch
patch < netbsd-expiretable-0.6.patch
cd expiretable-0.6
make
make install
```

Options de lancement :

- -a ancre : spécifie l'ancre contenant la table (privée).
- -d : lancement en tâche de fond. Implique l'option -p.
- -n : indique seulement les adresses qui vont être effacées sur la sortie standard mais ne procède pas à cette suppression.
- -p : modifie le comportement normal du programme qui, par défaut, est de quitter. Il s'exécute ainsi périodiquement (fonction de la durée fournie).
- -t durée : définit la durée de validité maximale d'une entrée dans cette table. Celle-ci peut être exprimée en secondes ou à l'aide de suffixes d (jour), h (heure), m (minute) ou s (seconde).
- -v : sortie verbeuse. Doublez la pour obtenir un maximum d'informations.

Exemple d'utilisation : supprimer régulièrement (exécution en tâche de fond) toutes les entrées datant de plus d'une heure de la table utilisateurs_authpf.

```
expiretable -d -p -t 1h utilisateurs_authpf
```

Retrouvez l'article de julp en ligne : Lien89

Les livres Linux

Les clés de l'administration système sous Linux

Cet ouvrage s'adresse à tous ceux qui administrent un système Linux.
Les livres traitant de l'administration système sont assez prévisibles. Ils vous expliquent comment gérer les utilisateurs, les systèmes de fichiers, les périphériques, les processus, les imprimantes, les réseaux, etc.

En revanche, ils ne vous disent pas ce que vous devez faire lorsque de nouveaux problèmes surviennent.

Si votre site web devient populaire, vous devrez vous documenter très vite sur les serveurs proxy, les différents niveaux de cache, la répartition de charge, l'authentification répartie, ainsi que d'autres sujets complexes.

Si vous avez ajouté une base de données, vous devrez rapidement la faire évoluer et apprendre à éviter les attaques par injection SQL.

Du jour au lendemain, des sites deviennent critiques et vous devez alors effectuer des sauvegardes à chaud sur des systèmes fonctionnant vingt-quatre heures sur vingt-quatre et sept jours sur sept.

Critique du livre par Pierre Schwartz

Ce livre a pour but de fournir aux administrateurs de systèmes Linux en détresse une panoplie de tutoriels pour leur apprendre à configurer un certain nombre de services parmi lesquels on peut citer Apache, Bind, MySQL, PHP et Samba. Chacun de ces tutoriels est traité de manière assez poussée en abordant les options de configuration avancées de ces outils, en donnant toutes les commandes nécessaires et tous les fichiers de configuration à modifier.

Ne comptez cependant pas sur les explications pédagogiques ou sur les détails du système Linux proprement dits, vous seriez déçu. Bien que chaque tutoriel soit précédé de quelques explications formelles le remettant dans son contexte d'utilisation, le livre passe très vite à l'aspect pratique en donnant la liste des

instructions à taper sans beaucoup plus d'explications.

Ce livre apparait clairement destiné aux administrateurs souhaitant collecter des procédures claires pour réaliser des installations de serveurs. Curieux du système Linux, passez votre chemin. C'est le reproche que je fais à ce livre, il n'apporte pas réellement de valeur ajoutée par rapport à des tutoriels disponibles sur Internet. Certes on y trouve des procédures avancées et robustes mais il manque de profondeur et de justifications sur les choix effectués. Il est donc nécessaire de bien maîtriser l'environnement Linux et d'avoir déjà manipulé les différents outils étudiés ici (avec ou sans succès) pour tirer pleinement profit de cet ouvrage.

Le noyau Linux

Vous êtes-vous déjà demandé pourquoi Linux était aussi efficace ? Voulez-vous savoir si ses performances seront opérantes avec votre application préférée ? Avez-vous déjà jeté un œil au code source du noyau ? Souhaitez-vous simplement comprendre comment fonctionne un système d'exploitation moderne ? Si vous acquiescez à chacune de ces questions, alors cet ouvrage est fait pour vous. La lecture de cette troisième édition vous éclairera sur ce qui fait de Linux l'un des meilleurs systèmes et comment il fournit une réponse efficace au défi de l'ordonnancement de processus, de l'accès aux fichiers et à la gestion de la mémoire dans une multiplicité d'environnements. La plupart des structures de données importantes, de nombreux algorithmes ou astuces de programmation en usage dans le noyau sont étudiés ; dans de nombreux cas, les fragments de codes pertinents sont analysés ligne par ligne. Par ailleurs, de nombreuses discussions relatives à Intel enrichissent cette nouvelle édition qui couvre les noyaux 2.6. Les auteurs introduisent chaque chapitre en expliquant l'importance et l'interaction entre le noyau et les utilitaires familiers des utilisateurs et des programmeurs. Cet ouvrage ne s'adresse pas exclusivement aux administrateurs système ou aux programmeurs, mais aussi aux étudiants et aux passionnés qui souhaitent comprendre comment fonctionnent réellement les choses à l'intérieur de la machine. Le noyau Linux est une visite guidée à travers des milliers de lignes de code : en route pour l'exploration !

Critique du livre par Mathieu Dalbin

Ce livre a été écrit à partir, dans un premiers temps, de supports de cours et a été par la suite enrichi petit à petit par les deux auteurs en fonction des évolutions du noyau Linux. Cette évolution a permis d'aboutir à la troisième édition de cet ouvrage, qui traite désormais de la version 2.6 du noyau Linux. Ce livre s'adresse en priorité aux personnes souhaitant découvrir comment fonctionne le noyau Linux : ordonnancement des processus, gestion de la mémoire, communications entre processus, systèmes de fichiers... Cela dit, le livre ne s'adresse pas aux débutants, car de bonnes bases de programmation et des connaisances en architecture des ordinateurs (x86 principalement) sont nécessaires pour aborder pleinement le contenu technique du livre.

En effet, on retrouve des pages d'explication sur le fonctionnement interne du noyau et une liste de détails techniques, qui apparaît être très exhaustive : rien n'est oublié ! Des bouts de codes en C et en assembleur, ainsi que des schémas explicatifs, agrémentent un texte qui se lit aisément.

Au sommaire de cette exploration du noyau, on retrouve donc les éléments classiques mais indispensables de tout noyau de système d'exploitation : gestion des processus et odonnancement, gestion de la mémoire et pagination, communication entre processus, gestion des interruptions, synchronisation, modules, systèmes de fichiers, et bien d'autres chapitres. Ce livre semble indispensable pour quiconque s'intéressant à la structure du noyau Linux et voulant en modifier les sources. Ce livre peut également être utile aux programmeurs de modules ou de pilotes de périphériques par exemple, tant son exhaustivité est exemplaire. Un "Must" !

Retrouvez ces critiques de livre sur la page livres Linux : Lien90

Les derniers tutoriels et articles

Manipuler des fichiers XML en VBScript avec XPath

Cet article va vous apprendre à manipuler des fichiers XML en Visual Basic Script avec Xpath.

1. Introduction

Dans cet article, nous allons commencer par apprendre à lire un fichier XML existant et à en extraire les informations dont on a besoin. Ensuite, nous allons nous pencher sur la création de documents XML. Et enfin, nous allons voir comment faire pour modifier un fichier XML existant.

Pour cela nous allons utiliser l'api DOM de Microsoft (Microsoft DOM (Lien91)). Il s'agit d'une implémentation de la recommandation définie par le W3C qui permet d'accéder au contenu d'un document et de le modifier. Le DOM est surtout utilisé pour modifier des documents XML ou pour accéder au contenu de pages web.

Pour le parcours du document DOM, nous allons utiliser XPath. XPath est une syntaxe permettant de désigner une portion précise d'un fichier XML.

2. Lecture de fichiers XML

Voici le document XML sur lequel nous allons travailler :

personnes.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<personnes>
	<personne age="49">
		<nom>Baud</nom>
		<prenom>Georges</prenom>
		<etat>Marié</etat>
		<enfants>
			<enfant>
				<nom>Tiop</nom>
				<prenom>Elisabeth</prenom>
			</enfant>
		</enfants>
	</personne>
	<personne age="22">
		<nom>Trinzka</nom>
		<prenom>Judith</prenom>
		<etat>Célibataire</etat>
	</personne>
	<personne age="88">
		<nom>Godoh</nom>
		<prenom>Madeleine</prenom>
		<etat>Veuve</etat>
		<enfants>
			<enfant>
				<nom>Godoh</nom>
				<prenom>Jean-
Marie</prenom>
			</enfant>
			<enfant>
				<nom>Godoh</nom>
				<prenom>Etienne</prenom>
			</enfant>
			<enfant>
				<nom>Swoti</nom>
				<prenom>Julienne</prenom>
			</enfant>
		</enfants>
	</personne>
</personnes>
```

C'est donc simplement une liste de personnes. Chaque personne a un âge, un état, un nom et un prénom et peut éventuellement avoir des enfants.

On va maintenant passer à la lecture du fichier XML. Pour commencer, il nous faut utiliser un parseur. Pour cela, nous allons utiliser le parseur XMLDOM de Microsoft.

La première chose à faire dans notre code est donc d'initialiser un parseur :

```
Set xmlDoc = CreateObject("Microsoft.XMLDOM")
```

Et tout à la fin de notre script, il ne faut pas oublier de détruire notre objet :

```
Set xmlDoc = Nothing
```

Il nous faut maintenant ouvrir notre fichier. On va également indiquer au parseur de charger tout le fichier en mémoire avant de commencer à le parser :

```
Set xmlDoc = CreateObject("Microsoft.XMLDOM")
xmlDoc.Async = "false"
xmlDoc.Load("personnes.xml")
```

On a donc juste passé le parseur en mode synchrone pour qu'il charge tout le fichier en mémoire avant de le traiter et ensuite on a ouvert le fichier avec la méthode Load.

Pour l'exemple, on va juste créer un petit programme qui va lire le document XML et afficher les personnes dans une boîte de dialogue.

On va maintenant récupérer toutes les personnes contenues dans le fichier XML. Pour cela, on va utiliser la méthode selectNodes à laquelle on passe une requête XPath. La requête XPath est simple, on va rechercher tous les éléments personne dans la balises personnes :

```
'On récupère tous les noeuds personnes
'à l'intérieur d'un noeud personnes
```

```
For Each personneElement In
xmlDoc.selectNodes("/personnes/personne")

Next
```

On a donc bouclé sur tous les éléments personnes que l'on a trouvés dans l'élément racine.

On va maintenant récupérer les informations principales (nom, prénom, état) et les afficher avec la méthode MsgBox :

```
'On récupère les informations sur la personne
nom =  personneElement.selectSingleNode("nom").text
prenom =
personneElement.selectSingleNode("prenom").text
etat =  personneElement.selectSingleNode("etat").text

MsgBox "Nom : " & nom & vbcrlf & _
               "Prénom : " & prenom & vbcrlf & _
               "Etat civil : " & etat
```

Ce qui devrait vous afficher toutes les personnes dans des boites de dialogue. Pour récupérer le contenu d'un élément, on a utilisé la méthode selectSingleNode sur le noeud XML qui permet de récupérer un noeud unique depuis le noeud courant et on a ensuite récupérer la valeur du noeud avec l'attribut text.

On va maintenant récupérer l'âge de la personne et ajouter ça dans notre fenêtre :

```
'On récupère les informations sur la personne
nom =  personneElement.selectSingleNode("nom").text
prenom =
personneElement.selectSingleNode("prenom").text
etat =  personneElement.selectSingleNode("etat").text
age = personneElement.getAttribute("age")

MsgBox "Nom : " & nom & vbcrlf & _
               "Prénom : " & prenom & vbcrlf & _
               "Etat civil : " & etat & vbcrlf & _
               "Age : " & age
```

Cette fois, c'est donc la méthode getAttribute que l'on a utilisé sur notre noeud. Enfin, on va maintenant voir comment faire pour récupérer les enfants d'une personne :

```
'On récupère tous les noeuds enfant à l'intérieur d'une
balise enfants dans le noeud personne courant
Set enfants =
personneElement.selectNodes("enfants/enfant")

'S'il y a des enfants
If enfants.length > 0 Then
       text = prenom & " " & nom & " a des enfants : "

       'On boucle sur tous les enfants
       For Each enfantElement In enfants
              nomEnfant =
enfantElement.selectSingleNode("nom").text
              prenomEnfant =
enfantElement.selectSingleNode("prenom").text

              text = text & vbcrlf & " - " &
prenomEnfant & " " & nomEnfant
       Next

       MsgBox text
Else
       MsgBox prenom & " " & nom & " n'a pas d'enfants"
End If
```

On a donc commencé par récupérer tous les éléments enfant de la personne et s'il y en a, on les a parcourus pour les ajouter et pour enfin, les afficher. S'il n'y en a pas, on affiche tout simplement un message d'erreur.

Voilà. On a donc réussi à parser notre fichier XML et à en extraire les informations nécessaires. La lecture de fichier repose toujours sur le même principe. On récupère des noeuds, leurs enfants, leur contenu et leurs attributs.

Maintenant que l'on a vu comment lire un fichier XML, on va voir comment est-ce qu'on peut en écrire un.

3. Ecriture de fichier XML

On va maintenant faire le travail inverse de l'exercice précédent, on va partir des données contenues dans notre programme VBS pour écrire un fichier personnes2.xml.

Voilà la forme sous laquelle on a nos données :

```
personnes = array()
Redim personnes(1)

georges = array()
Redim georges(4)

georges(0) = "Baud"
georges(1) = "Georges"
georges(2) = "Marié"
georges(3) = 49

enfants = array()
Redim enfants(0)

elisabeth = array()
Redim elisabeth(1)

elisabeth(0) = "Tiop"
elisabeth(1) = "Elisabeth"

enfants(0) = elisabeth

georges(4) = enfants

personnes(0) = georges

judith = array()
Redim judith(4)

judith(0) = "Trinzka"
judith(1) = "Judith"
judith(2) = "Célibataire"
judith(3) = 22
judith(4) = array()

personnes(1) = judith
```

Et on va transformer en un XML de la même forme que celui qu'on lisait au chapitre précédent

Pour l'écriture, on va également devoir initialiser et configurer notre parseur :

```
Set xmlDoc = CreateObject("Microsoft.XMLDOM")

Set oCreation =
xmlDoc.createProcessingInstruction("xml",
```

```
"version='1.0' encoding='ISO-8859-1'")
xmlDoc.insertBefore oCreation,
xmlDoc.childNodes.Item(0)
```

Les 2 lignes qui suivent l'initialisation permettent de générer l'entête XML.

Ensuite, on va créer la racine et l'ajouter au document :

```
Set root = xmlDoc.createElement("personnes")

xmlDoc.appendChild(root)
```

On a donc utilisé la méthode createElement pour créer un nouveau noeud et la méthode appendChild pour ajouter ce noeud à la racine du document.

On va maintenant parcourir notre tableau personnes pour créer un élément personne pour chacun des éléments du tableau :

```
For Each personne In personnes
	Set personneElement =
xmlDoc.createElement("personne")
	root.appendChild(personneElement)
Next
```

Cette fois, on ajouté les éléments dans le noeud racine.

On va maintenant ajouter les informations principales sur la personne :

```
Set personneElement = xmlDoc.createElement("personne")

Set nomElement = xmlDoc.createElement("nom")
nomElement.Text = personne(0)
personneElement.appendChild(nomElement)

Set prenomElement = xmlDoc.createElement("prenom")
prenomElement.Text = personne(1)
personneElement.appendChild(prenomElement)

Set etatElement = xmlDoc.createElement("etat")
etatElement.Text = personne(2)
personneElement.appendChild(etatElement)

root.appendChild(personneElement)
```

Le principe reste toujours le même, on crée un élément et on l'ajoute à l'élément parent.

Passons maintenant à l'ajout de l'attribut age :

```
personneElement.setAttribute "age", personne(3)
```

Rien de bien méchant de ce coté-là. Pour les enfants, non plus, rien de difficile :

```
If UBound(personne(4)) > -1 Then
	Set enfantsElement =
xmlDoc.createElement("enfants")

	For Each enfant In personne(4)
		Set enfantElement =
xmlDoc.createElement("enfant")

		Set nomElement =
xmlDoc.createElement("nom")
		nomElement.Text = enfant(0)
		enfantElement.appendChild(nomElement)

		Set prenomElement =
xmlDoc.createElement("prenom")
		prenomElement.Text = enfant(1)
		enfantElement.appendChild(prenomElement)


	enfantsElement.appendChild(enfantElement)
	Next

	personneElement.appendChild(enfantsElement)
End If
```

S'il y a des enfants, on crée donc un élément enfants auquel on ajoute des éléments enfant pour chaque enfant de la personne et enfin on ajoute un nom et un prénom à chaque enfant.

On va maintenant sauvegarder notre fichier.

```
xmlDoc.save ("personnes2.xml")
```

Et voilà le résultat :

personnes2.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<personnes><personne
age="49"><nom>Baud</nom><prenom>Georges</prenom><etat>M
arié</etat><enfants><enfant><nom>Tiop</nom><prenom>Elis
abeth</prenom></enfant></enfants></personne><personne
age="22"><nom>Trinzka</nom><prenom>Judith</prenom><etat
>Célibataire</etat></personne></personnes>
```

Ce fichier XML bien que tout à fait valide n'est pas très lisible. On va voir dans le chapitre suivant comment faire pour le rendre plus lisible.

3.1. Indenter le fichier XML

La méthode à utiliser est malheureusement bien plus compliquée cette fois. Il faut remplacer xmlDoc.Save par :

```
set rdr = CreateObject("MSXML2.SAXXMLReader")
set wrt = CreateObject("MSXML2.MXXMLWriter")
Set oStream = CreateObject("ADODB.STREAM")
oStream.Open
oStream.Charset = "ISO-8859-1"

wrt.indent = True
wrt.encoding = "ISO-8859-1"
wrt.output = oStream
Set rdr.contentHandler = wrt
Set rdr.errorHandler = wrt
rdr.Parse xmlDoc
wrt.flush

oStream.SaveToFile "personnes2.xml", 2

Set rdr = Nothing
Set wrt = Nothing
```

Ce qu'on fait dans ce code, c'est qu'on ouvre un reader SAX qui va parser notre fichier XML et on donne le contenu de la lecture à un MXXMLWriter qui va nous permettre d'écrire notre XML avec les balises indentées et enfin, on configure le stream de sortir du writer on lui dit d'écrire dans le fichier de notre choix.

Ce qui nous donne comme fichier en sortie :

personnes2.xml

```
<?xml version="1.0" encoding="ISO-8859-1"
standalone="no"?>
<personnes>
	<personne age="49">
		<nom>Baud</nom>
		<prenom>Georges</prenom>
		<etat>Marié</etat>
		<enfants>
			<enfant>
				<nom>Tiop</nom>
				<prenom>Elisabeth</prenom>
			</enfant>
		</enfants>
	</personne>
	<personne age="22">
		<nom>Trinzka</nom>
		<prenom>Judith</prenom>
		<etat>Célibataire</etat>
	</personne>
</personnes>
```

Il est donc directement beaucoup plus lisible.

4. Modification

On va maintenant voir comment faire pour modifier un fichier XML existant. En fait, ce n'est pas bien compliqué et on a déja tout vu. Il suffit en fait de lire le fichier XML, de le modifier avec les commandes vues dans la partie d'écriture et de le sauvegarder ensuite.

Pour l'exemple, on va modifier les âges des personnes. On va rajouter une année à chacune des personnes dans le XML.

On va l'ouvrir de la même manière que pour une lecture normale :

```
Set xmlDoc = CreateObject("Microsoft.XMLDOM")
xmlDoc.Async = "false"
xmlDoc.Load("personnes.xml")
```

Ensuite, on va boucler sur toutes les personnes et leur rajouter une année :

```
'On récupère tous les noeuds personnes à l'intérieur
d'un noeud personnes
For Each personneElement In
xmlDoc.selectNodes("/personnes/personne")
	'On récupère l'âge de la personne
	age = personneElement.getAttribute("age")

	'On lui rajoute un an
	personneElement.setAttribute "age", age + 1
Next
```

Et finalement, on va sauvegarder notre fichier. Vous pouvez choisir la méthode que vous préférez pour le sauvegarder. Pour l'exemple, on va prendre la méthode avec indentation :

```
set rdr = CreateObject("MSXML2.SAXXMLReader")
set wrt = CreateObject("MSXML2.MXXMLWriter")
Set oStream = CreateObject("ADODB.STREAM")
oStream.Open
oStream.Charset = "ISO-8859-1"

wrt.indent = True
wrt.encoding = "ISO-8859-1"
wrt.output = oStream
Set rdr.contentHandler = wrt
Set rdr.errorHandler = wrt
rdr.Parse xmlDoc
wrt.flush

oStream.SaveToFile "personnes.xml", 2

Set rdr = Nothing
Set wrt = Nothing
Set xmlDoc = Nothing
```

Ce qui vous donnera comme fichier XML :

personnes.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<personnes>
	<personne age="50">
		<nom>Baud</nom>
		<prenom>Georges</prenom>
		<etat>Marié</etat>
		<enfants>
			<enfant>
				<nom>Tiop</nom>
				<prenom>Elisabeth</prenom>
			</enfant>
		</enfants>
	</personne>
	<personne age="23">
		<nom>Trinzka</nom>
		<prenom>Judith</prenom>
		<etat>Célibataire</etat>
	</personne>
	<personne age="89">
		<nom>Godoh</nom>
		<prenom>Madeleine</prenom>
		<etat>Veuve</etat>
		<enfants>
			<enfant>
				<nom>Godoh</nom>
				<prenom>Jean-
Marie</prenom>
			</enfant>
			<enfant>
				<nom>Godoh</nom>
				<prenom>Etienne</prenom>
			</enfant>
			<enfant>
				<nom>Swoti</nom>
				<prenom>Julienne</prenom>
			</enfant>
		</enfants>
	</personne>
</personnes>
```

Comme vous le voyez, une fois que vous savez lire et écrire dans un fichier XML, la modification n'est plus qu'un jeu d'enfant.

On va prendre un deuxième exemple : Mme Madeleine Godoh a renié ses enfants pour une raison quelconque, on va donc devoir les supprimer du fichier XML.

Je ne vais montrer que la partie de modification le reste étant le même code que pour le premier exemple :

```
'On récupère tous les noeuds enfant à l'intérieur d'une
balise enfants
'se trouvant dans un noeud personne qui a comme nom
Godoh et comme prénom Madeleine
xmlDoc.selectNodes("/personnes/personne[nom='Godoh' and
prenom='Madeleine']/enfants/enfant").removeAll
```

Cette fois, on voit toute la puissance de XPath. Il nous suffit d'une

seule ligne pour supprimer les enfants d'une personne. On a donc mis la recherche directement dans la requête XPath et on a ensuite demandé les enfants pour la personne correspondante à a la requête et enfin on a supprimé tous ces éléments.

Et voici donc le résultat :

personnes.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<personnes>
	<personne age="50">
		<nom>Baud</nom>
		<prenom>Georges</prenom>
		<etat>Marié</etat>
		<enfants>
			<enfant>
				<nom>Tiop</nom>
				<prenom>Elisabeth</prenom>
			</enfant>
		</enfants>
	</personne>
	<personne age="23">
		<nom>Trinzka</nom>
		<prenom>Judith</prenom>
		<etat>Célibataire</etat>
	</personne>
	<personne age="89">
		<nom>Godoh</nom>
		<prenom>Madeleine</prenom>
		<etat>Veuve</etat>
		<enfants>
		</enfants>
	</personne>
</personnes>
```

5. Conclusion

Voilà, vous savez maintenant comment lire et écrire dans des fichiers XML en Visual Basic Script. J'espère que cet article vous aura été utile.

Si vous voulez en savoir plus sur XPath, je vous conseille de lire cet article (Lien92) qui montre une grande quantité d'exemples de requêtes XPath.

Retrouvez l'article de Baptiste Wicht en ligne : Lien93

Les derniers tutoriels et articles

Delphi 2007 - Découverte de l'IDE

Cet article vous propose de découvrir les principales nouveautés de l'IDE de Delphi 2007 par rapport à Delphi 7.

1. Introduction

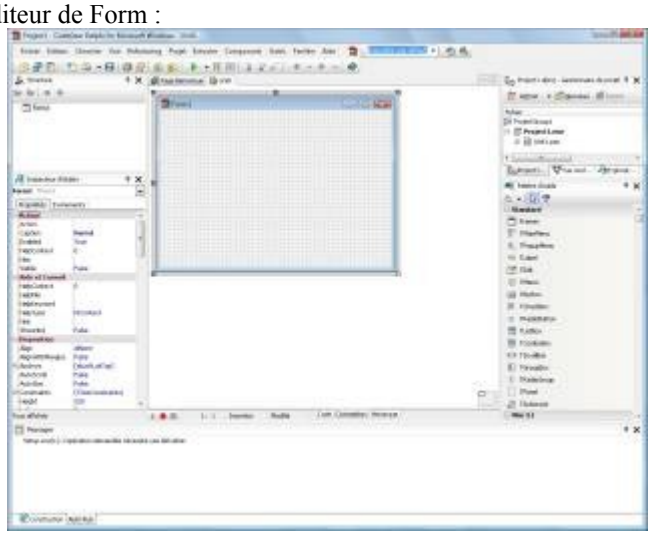
1.1. Premier démarrage

L'éditeur dans son intégralité se présente en un seul bloc, divisé en sous-bloc contenant des informations.
Au premier démarrage de Delphi 2007, la page de Bienvenue est affichée sous form de page Web locale, affichant plusieurs informations utiles telle qu'une aide pour la prise en main, ou encore les actualités courantes concernant Delphi.

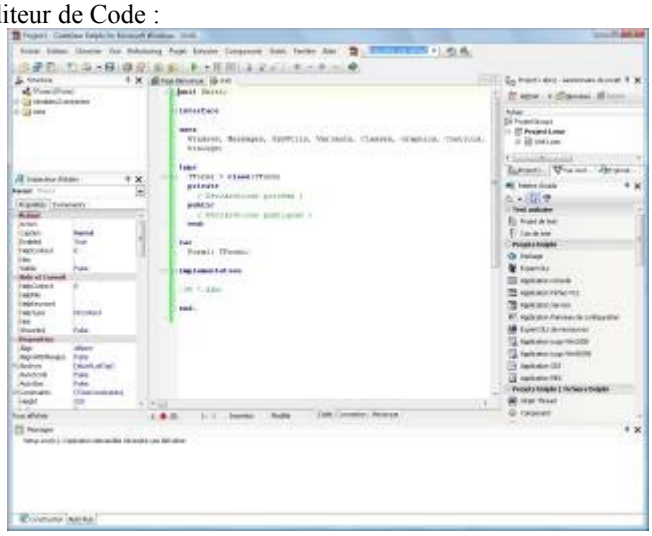
1.2. Nouveau projet

Afin de visualiser l'environnement, j'ai créé un projet "classique": "Application Fiches VCL"
Ce type projet est soit accessible par le menu "Fichier" soit sur la palette de composant (que nous verrons un peu plus tard).

Editeur de Form :

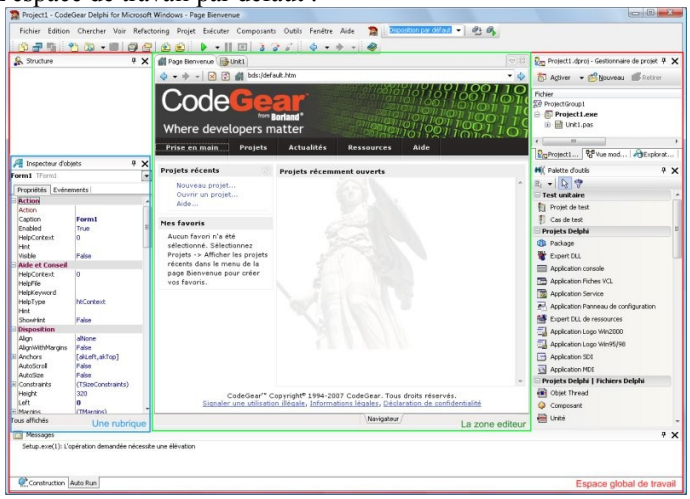


Editeur de Code :



2. L'espace de travail

L'espace de travail par défaut :



L'espace de travail est une fenêtre globale partagée par:

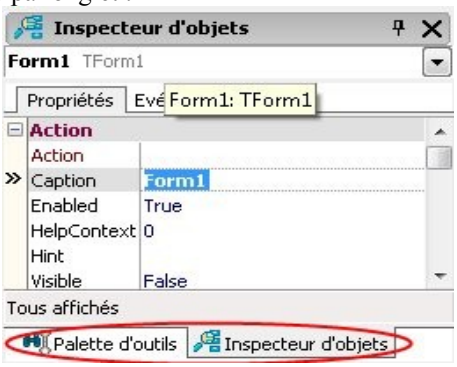
- le menu (potentiellement flottant)
- les barres d'outils (potentiellement flottantes)
- la fenêtre principale, sur laquelle sont répartis:
  - L'éditeur de code en multi-onglets
  - Les rubriques (inspecteur d'objets, gestionnaire de projet, ... mais aussi la palette d'outils)

Chaque rubrique est contenue dans une zone et peut être affichée de 3 manières différentes

toute seule, occupant une zone de la fenêtre globale :

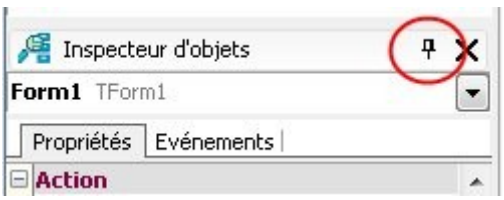
en fenêtre flottante indépendante :

avec d'autre rubriques, la zone est partagée et la rubrique est sélectionnée par onglet :



Le déplacement de ces rubriques se fait par glisser/déposer à la souris et avec des "aimants" visuels un peu partout.
Les aimants visuels sont des zones bleues affichées pendant le déplacement représentant la rubrique, ils montrent où la rubrique sera déposée.

Lorsque une rubrique se trouve dans une zone de la fenêtre globale, il est possible "d'agrafer" ou non cette rubrique.
La rubrique est ainsi affichée en permanence.
Quand il n'y a plus d'agrafe (sur un simple clic), cette zone se replie automatiquement sur le bord de l'écran pour gagner un espace précieux, en laissant une zone accessible à la souris pour déplier à nouveau la zone à la demande.



3. Particularités de la zone Editeur

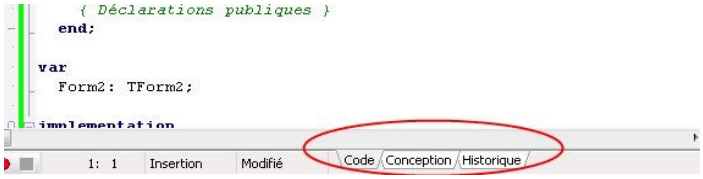
3.1. Zone éditeur

L'éditeur de code occupera la place disponible restante, chaque fichier est ouvert dans un onglet supplémentaire.

L'onglet courant affiché contient 3 sous-onglets permettant d'afficher successivement:

- 'Code' la fenêtre de code
- 'Conception' la fenêtre de conception des Forms (TForm, TFrame...)
- 'Historique' qui permet de retrouver son code et/ou sa conception des versions précédentes, très utile !

Navigation entre Editeur de Form, de code et historique par onglet

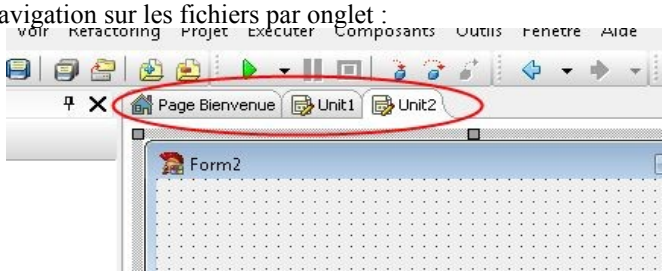


A noter, que F12 permet toujours de "switcher" du code à la fenêtre de conception, mais contrairement à Delphi 7 et par défaut, c'est soit l'un, soit l'autre.

3.2. Editeur de code

Chaque nouveau fichier, est ouvert dans la zone éditeur, dans un nouvel onglet.

Navigation sur les fichiers par onglet :



La gouttière de l'éditeur de code affiche dorénavant des informations utiles en temps réel.

- le numéro de la ligne
- un point rouge pour un point d'arrêt
- un trait épais vertical, de couleur verte lorsque le code n'a pas été modifié depuis la dernière sauvegarde, jaune sinon
- un système de volet pliant/dépliant pour masquer/cacher des portions de codes/procédures/classes. Il est possible de définir ses propres zones à masquer/afficher

Gouttière et section pliable :

Par défaut, les sections pliables sont les procédures et définitions de classes, mais il est tout à fait possible de définir soit même les sections pliables par des directives de compilation.

Fermeture visuelle des parenthèses/crochets
Visuellement indispensable pour éviter de passer du temps à compter les parenthèses !

Fermetures des parenthèses :

```
procedure TForm3.BitBtn1Click(Sender: TObject);
var
   i, j, k: Integer;
begin
     i := ( (j + k) + ( j * k) ) * i
end;
```

Assistants dans l'écriture du code.
Voici un petit aperçu des fonctions pour assister l'écriture de code. Ces fonctions peuvent paraître déroutantes au premier abord, mais sont très utiles après un temps d'adaptation.

Code en erreur surligné en rouge instantanément :

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  ShowMessage( 'Hello world!' );
  ShoMessage(
end;
```

Surlignage temps réel d'une erreur

La saisie de 'for' génère automatiquement des zones tabulables pour saisir les éléments encadrés, et cerise sur le gateau, la variable I est ajoutée automatiquement dans les variables locales :

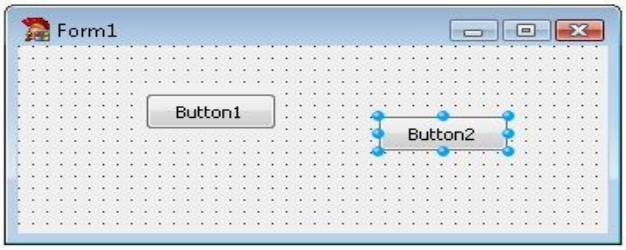
```
procedure TForm1.Button1Click(Sender: TObject);
begin
  ShowMessage( 'Hello world!' );
  for I := 0 to List.Count - 1 do

end;
```

3.3. Editeur de Form

Plusieurs améliorations visuelles ont été apportées lors de la conception d'une forme.

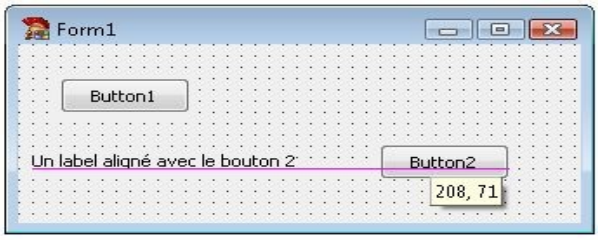
En voici quelques unes:

Les éléments sélectionnés sont mis en évidence par des points bleus bien visibles :



Pendant le déplacement d'un composant, une ligne bleue est affichée et montre l'alignement éventuel avec les composants voisins (ici entre une zone texte et le bouton) :

Même chose avec une ligne magenta mais... en se basant sur le texte contenu dans le composant! :



Mise en évidence et aimantation d'un élément par rapport aux bord de la Form, symbolisé par une ligne noire :

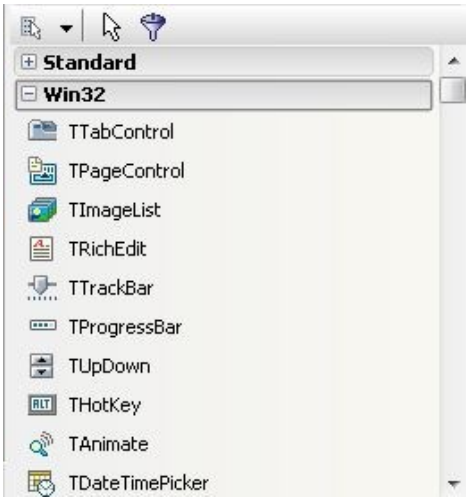
Malheureusement, aucune modification au niveau de l'ordre de tabulation des composants n'a été apportée.
L'ordre des composants est toujours défini la liste des composants visuels, dans laquelle il faut classer les composants un à un par décalage.

4. La palette d'outils

4.1. Disposition

La palette d'outils est une rubrique comme les autres. Le contenu peut donc être déplacé à peu près n'importe ou.
Les composants sont toujours classés par section.
La grande nouveauté est la possibilité de fortement personnaliser l'affichage:

Aspect standard des composants. Les sections peuvent être pliées ou dépliées selon les besoins :



Il est toujours possible de se créer des catégories personnelles et d'y placer ses composants usuels.
Il y a également un système de filtre par catégorie, qui permet de ne pas afficher certaines catégories.
Pour avoir une vision globale des composants disponible, les catégories peuvent rester pliées, se déplier le temps de choisir un composant, et se replier automatiquement.

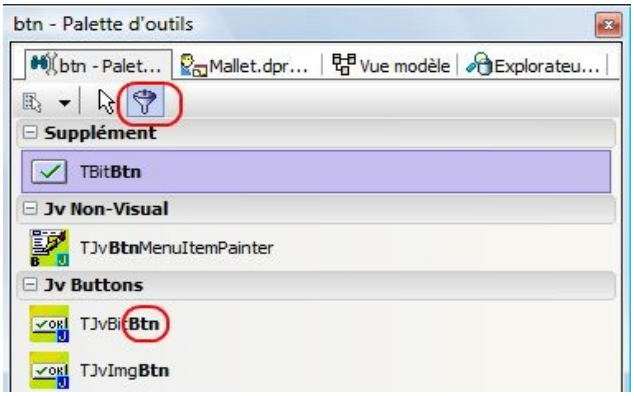
4.2. Recherche de composants

Au premier abord, vu la multitude de composants qu'il y a, on peut douter de la rapidité de recherche des composants, qu'ils soient usuels ou non.
Heureusement, la fonction Filtre permet de retrouver très

rapidement un composant en tapant quelques lettres du composant.

Exemple de recherche des composants boutons, en saisissant simplement 'btn' dans la palette:

Filtre en tapant 'btn' :

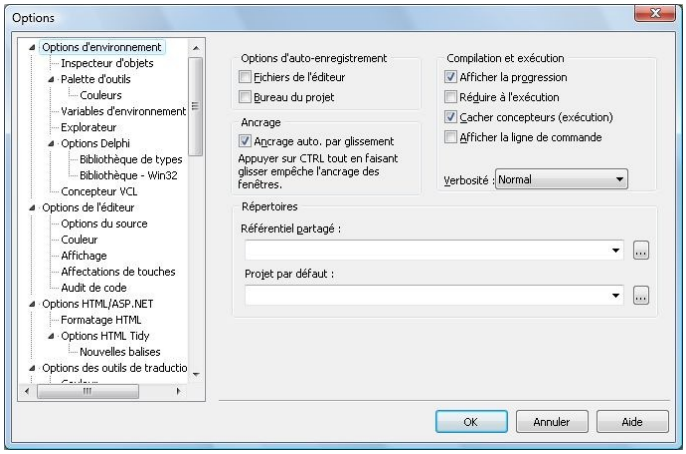


5. Autres aspects

5.1. Fenêtre de configuration

Par rapport à Delphi 7, beaucoup de changements dans le système de modification du paramétrage. Les options se sont étoffées au fil du temps, la navigation se fait maintenant de manière hiérarchique plutot que par onglet.
Malheureusement, bien qu'un "splitbar" vertical permet d'agrandir la zone hiérachique des options, la fenêtre dans sa globalité n'est pas redimensionnable ce qui a pour conséquence que l'on se sent un peu "étriqué" par moment.

Fenêtre des options :



A noter que pour la majorité des rubriques, le menu contextuel sur celles-ci permet d'accéder directement aux options concernées, sans passer par la fenêtre globales des options.

5.2. Aide en ligne

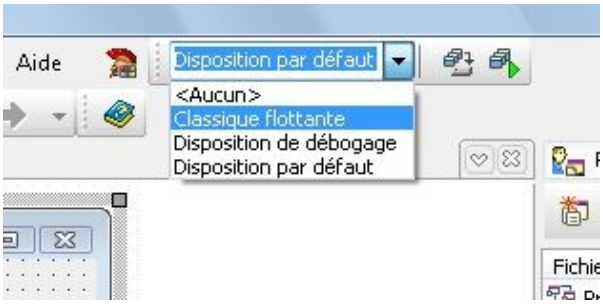
L'aide en ligne a été totalement modifiée par rapport à Delphi7.
L'aspect est plus moderne, aux nouvelles normes de Windows, et surtout connectée par Internet au site de Codegear.
Cependant, je dois avouer que pour le moment, elle ne m'a pas apporté une grande satisfaction quant à son contenu, mais je pense qu'au fil du temps, cela évoluera.

6. Comment retrouver un environnement avec le style de Delphi 7

Pour les nostalgiques de la gamme Delphi 7 (et plus ancien...), il est tout à fait possible de retrouver un environnement assez similaire à nos chères habitudes. 2 étapes principales sont nécessaires:

Rendre les fenêtres flottantes

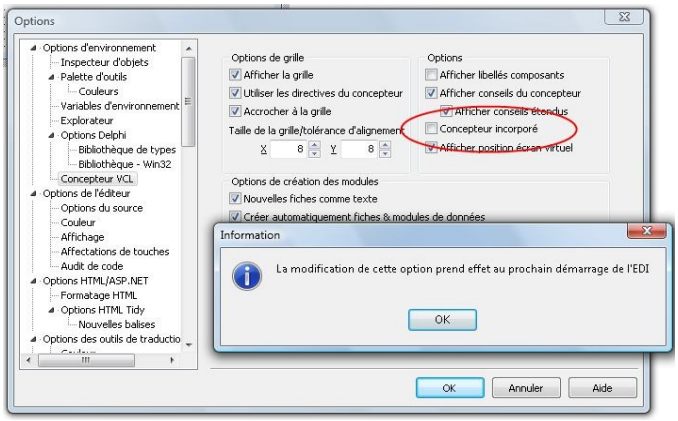
Disposition flottante :



Choisir la disposition "Classique flottant" dans la liste des dispositions.
A noter que la gestion des dispositions se trouve dans le menu "Voir / Bureaux"
A partir de là, chaque Zone se retrouve dans une fenêtre flottante, il est tout à fait possible d'avoir plusieurs catégories dans une seule zone, donc dans une seule fenêtre flottante.

"Libérer" l'éditeur de form

Libérer l'éditeur de form en fenêtre flottante :



Malgrès les fenêtres flottantes, l'éditeur de Code et l'éditeur de Form ne peuvent être affichés simultanément, c'est soit l'un soit l'autre.
Or, il peut être très utile de visualiser la Form (et d'utiliser les infos dynamiques au survol de souris sur un composant), pendant l'écriture du code.
Pour cela, dans la fenêtre des options, il faut décocher la case "Concepteur incorporé" puis redémarrer Delphi 2007.

7. Conclusion

Beaucoup de changements par rapport à Delphi 7 au niveau de l'IDE, moins par rapport aux versions intérmédiaires (Delphi 2005, Delphi2006 et Turbo Delphi).
Mais globalement, avec un petit effort de remise à niveau sur nos habitudes, Delphi 2007 devient très vite agréable et rapide à utiliser.

Retrouvez l'article de TicTacToe en ligne : Lien94

Liens

Lien1 : http://nicolas-zozol.developpez.com/tutoriel/java/jtable/
Lien2 : http://www.eyrolles.com/Accueil/Livre/9782212117103/
Lien3 : http://hikage.developpez.com/interview/auteurs/spring-par-la-pratique/
Lien4 : http://pbnaigeon.developpez.com/tutoriel/PHP/apostrophe-guillemet/
Lien5 : http://frederic.bouchery.free.fr/?2004/08/10/9-Echo-Lapin-Ou-Tortue
Lien6 : http://fr2.php.net/manual/fr/language.types.string.php
Lien7 : http://www.faqts.com/knowledge_base/view.phtml/aid/1/fid/40
Lien8 : http://m-fernandez.developpez.com/articles/php/bench/
Lien9 : http://a-pellegrini.developpez.com/tutoriels/css/formulaire/exemple/form4.html
Lien10 : http://a-pellegrini.developpez.com/tutoriels/css/formulaire/exemple/form8.html
Lien11 : http://a-pellegrini.developpez.com/tutoriels/css/formulaire/
Lien12 : http://xhtml.developpez.com/faq/?page=html_generalites#html_doctypes
Lien13 : http://j-willette.developpez.com/tutoriels/javascript/formulaire/exemple/inputtext-1.html
Lien14 : http://j-willette.developpez.com/tutoriels/javascript/formulaire/exemple/inputtext-2.html
Lien15 : http://j-willette.developpez.com/tutoriels/javascript/formulaire/exemple/radiobuttons.html
Lien16 : http://j-willette.developpez.com/tutoriels/javascript/formulaire/exemple/checkboxes.html
Lien17 : http://j-willette.developpez.com/tutoriels/javascript/formulaire/exemple/button.html
Lien18 : http://j-willette.developpez.com/tutoriels/javascript/formulaire/exemple/inputfile.html
Lien19 : http://j-willette.developpez.com/tutoriels/javascript/formulaire/exemple/textarea-1.html
Lien20 : http://j-willette.developpez.com/tutoriels/javascript/formulaire/exemple/textarea-2.html
Lien21 : http://j-willette.developpez.com/tutoriels/javascript/formulaire/exemple/select.html
Lien22 : http://j-willette.developpez.com/tutoriels/javascript/formulaire/
Lien23 : http://dotnet.developpez.com/faq/dotnet/
Lien24 : http://dotnet.developpez.com/cours/#a_debuter
Lien25 : http://dotnet.developpez.com/cours/#a_dotnet2
Lien26 : http://tahe.developpez.com/dotnet/csharp/
Lien27 : http://plasserre.developpez.com/vbintro.htm
Lien28 : http://rmdiscala.developpez.com/cours/
Lien29 : http://dotnet.developpez.com/livres/
Lien30 : http://msdn2.microsoft.com/fr-fr/netframework/aa569263.aspx
Lien31 : http://dotnet.developpez.com/outils/?page=editeurs
Lien32 : http://msdn2.microsoft.com/fr-fr/express/aa975050.aspx
Lien33 : http://dotnet.developpez.com/cours/#a_edi
Lien34 : http://www.mono-project.com/
Lien35 : http://dotnet.developpez.com/cours/#a_winforms
Lien36 : http://dotnet.developpez.com/cours/#a_aspnet
Lien37 : http://dotnet.developpez.com/cours/#a_ppc
Lien38 : http://dotnet.developpez.com/cours/#a_db
Lien39 : http://sharepoint.developpez.com/
Lien40 : http://dotnet.developpez.com/livres/#livresASPNET
Lien41 : http://dotnet.developpez.com/livres/?page=tous#L2746036649
Lien42 : http://dotnet.developpez.com/livres/?page=tous#L2100485121
Lien43 : http://dotnet.developpez.com/livres/?page=tous#L1904811213
Lien44 : http://www.microsoft.com/expression/products/overview.aspx?key=blend
Lien45 : http://msdn2.microsoft.com/fr-fr/express/aa718378.aspx
Lien46 : http://www.microsoft.com/downloads/details.aspx?familyid=6053C6F8-82C8-479C-B25B-9ACA13141C9E&displaylang=fr
Lien47 : http://broux.developpez.com/articles/csharp/silverlight/#L1
Lien48 : http://msdn2.microsoft.com/fr-fr/asp.net/bb330941.aspx
Lien49 : http://msdn2.microsoft.com/fr-fr/vcsharp/bb409645.aspx
Lien50 : http://msdn2.microsoft.com/fr-fr/vbasic/bb265238.aspx
Lien51 : http://msdn2.microsoft.com/fr-fr/teamsystem/bb383581.aspx
Lien52 : ftp://ftp-developpez.com/rmdiscala/livres/CsharpExos.pdf
Lien53 : http://www.silverlight.net/
Lien54 : http://broux.developpez.com/articles/csharp/silverlight/
Lien55 : http://broux.developpez.com/articles/csharp/chat-silverlight/
Lien56 : http://msdn2.microsoft.com/en-us/xna/default.aspx
Lien57 : http://creators.xna.com/
Lien58 : http://fearyourself.developpez.com/tutoriel/xna/
Lien59 : http://nicoboo.developpez.com/articles/xna/presentation/
Lien60 : http://cs-sdl.sourceforge.net/index.php/Main_Page
Lien61 : http://cs-sdl.sourceforge.net/index.php/Category:Tutorials
Lien62 : http://www.riemers.net/
Lien63 : http://creators.xna.com/Education/Samples.aspx
Lien64 : http://www.ziggyware.com/
Lien65 : http://nxengine.developpez.com/
Lien66 : http://www.developpez.net/forums/forumdisplay.php?f=589
Lien67 : http://chrisk.free.fr/cariboost2/crbst_7.html
Lien68 : http://www.ogre3d.org/wiki/index.php/OgreDotNet
Lien69 : http://irrlichtnetcp.sourceforge.net/index.php/Main_Page
Lien70 : http://www.developpez.net/forums/forumdisplay.php?f=13
Lien71 : http://www.developpez.net/forums/forumdisplay.php?f=66
Lien72 : http://www.developpez.net/forums/forumdisplay.php?f=49
Lien73 : http://www.developpez.net/forums/forumdisplay.php?f=578
Lien74 : http://dotnet.developpez.com/faq/
Lien75 : http://sqlserver.developpez.com/faq/
Lien76 : http://sharepoint.developpez.com/faq/
Lien77 : http://broux.developpez.com/articles/dotnet/bien-debuter-en-dotnet/

Lien78 : http://sourceware.org/pthreads-win32/
Lien79 : http://franckh.developpez.com/tutoriels/posix/pthreads/
Lien80 : http://doc.trolltech.com/
Lien81 : http://doc.trolltech.com/4.3/index.html
Lien82 : http://miles.developpez.com/tutoriels/cpp/qt/compilation/
Lien83 : http://qt.developpez.com/faq/
Lien84 : http://warin.developpez.com/access/variablestemporaires/
Lien85 : http://www.itpro.fr/article.asp?mag=3&th=9&ss=8&id=2064
Lien86 : http://download.microsoft.com/download/d/d/7/dd75ece7-83de-45da-8bb1-cb233decf595/BDMTDM.doc
Lien87 : http://sqlpro.developpez.com/cours/sqlserver/haute-disponibilite/
Lien88 : http://sqlpro.developpez.com/cours/sqlserver/transactions-imbriquees/
Lien89 : http://julp.developpez.com/bsd/packet-filter/tables/
Lien90 : http://linux.developpez.com/livres/
Lien91 : http://msdn2.microsoft.com/en-us/library/ms766487.aspx
Lien92 : http://jerome.developpez.com/xmlxsl/xpath/
Lien93 : http://baptiste-wicht.developpez.com/tutoriel/vbs/xml/
Lien94 : http://tictactoe.developpez.com/delphi/article/delphi2007/ide/