Developpez Magazine

Edition de Août-Septembre 2008.
Numéro 17.
Magazine en ligne gratuit.
Diffusion de copies conformes à l'original autorisée.
Réalisation : Alexandre Pottiez
Rédaction : la rédaction de Developpez
Contact : magazine@redaction-developpez.com

Index

Editorial

Pour ce mois de rentrée, n'oubliez pas de glisser dans votre cartable votre magazine favori.

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

La rédaction

Article Qt & 2D/3D/Jeux

Intégrer Ogre à Qt

Ce tutoriel est destiné à illustrer l'intégration du moteur 3D Ogre à Qt. Les concepts utilisés ici sont tout à fait utilisables pour d'autres moteurs (comme IrrLicht) et/ou pour d'autres toolkit graphique (comme WxWidget ou gtkmm).

par **Denys Bulant**
Page 49

Article Office

clGdiplus : Premiers pas

Découvrez les bases de l'utilisation de la classe ClGdiPlus.

par **Thierry Gasperment**
Page 34

Article complet: Introduction à Scala : Méthodes et Fonctions

Dans ce troisième volet de la série "Introduction à Scala", je vais présenter la notion de méthodes et fonctions de Scala, qui est bien plus puissante que celle dans Java, dans la mesure où les fonction dans Scala sont des objets à part entière, pouvant être stockées dans une variable, passées ou retournées par une autre méthode, etc.

1. Présentation

La définition d'une méthode en Scala se présente comme suit:

```
[private|protected] [override] def nom [paramètres]
[type de retour] [corps]
```

Ici, ce qui est entre [ ] dénote un élément optionnel.

Notez l'utilisation du mot clé **def** pour définir une fonction.
Dans Scala, et tout comme pour les définitions des classes, la visibilité par défaut est public, à moins qu'on précise explicitement la visibilité privée (via private) ou protégée (via protected).

Tout comme Java, les fonctions dans Scala sont virtuelles par défaut. Par contre, et à l'inverse de Java, pour redéfinir une méthode virtuelle, il faut le préciser explicitement avec le mot clé override, ce qui permet d'éviter quelques séances intensives de débogage pour comprendre l'origine de bugs bizarroïdes :) (C'est la même approche que celle prise par C#).

La partie paramètres d'une fonction est un peu différente de celle de Java et ce pour diverses raisons:
- Tout d'abord, à cause de la convention Pascal pour les déclarations (nom:Type)
- Mais aussi le fait que les parenthèses sont complètement optionnelles pour les méthodes sans arguments
- Et enfin, une fonction peut déclarer plusieurs listes de paramètres (j'y reviendrais plus tard)

J'ai déjà montré quelques exemples de méthodes avec les getters et les setters.

En voici une autre:

```
class Person(var age : Int) {
  def increaseAge(a : Int) {
    age = age+a;
  }
}
```

Cette méthode augmente l'age de la personne de n années.
Son invocation n'a rien d'exceptionnel, et se présente comme suit:

```
object ScalaMethods {
  def main(args : Array[String]) : Unit = {
    val p = new Person(25);
    p.increaseAge(12)
    println(p.age)

  }
}
```

Cette méthode montre entre autres comment le type de retour est optionnel. D'ailleurs, elle ne le déclare pas. La définition exacte aurait été:

```
def increaseAge(a : Int) : Unit = {
  age = age+a;
}
```

2. De l'objet au fonctionnel

Dans cette partie, je vais montrer l'aspect fonctionnel de Scala à travers un exemple où on part d'une implémentation à base des paradigmes de l'OO, puis en convertissant vers une approche plus fonctionnelle.
Comme le veut la tradition, le premier jour est réservé au Community One. Malheureusement je n'ai pas pu assister au keynote du matin, mon avion ayant atterri à San Francisco à 12h30 heure locale.

2.1. La version OO

Il s'agit tout simplement d'implémenter une méthode qui effectue un traitement d'une façon répétitive mais périodique.
Dans l'approche OO (et Java plus précisément), on commencerait par créer une interface qui définit une méthode.
Puis, on créerait une méthode qui prend cette interface comme paramètre ainsi que la période de répétition.

Voici une implémentation de cette approche en Scala :

```
trait Action {
  def doSomething();
}

object PingAction extends Action {
  def doSomething() = {
    println("ping")
  }
}

object ScalaMethods {
  def doPeriodically(action : Action, period : Int) =
{
    while(true){
      action.doSomething
      Thread.sleep(period);
    }
  }

  def main(args : Array[String]) : Unit = {
    doPeriodically(PingAction, 1000)
  }
}
```

J'ai défini un trait Action (la chose qui ressemble le plus à une interface dans Scala) avec une méthode *doSomething*.
J'ai ensuite crée une implémentation *PingAction* de ce trait qui ne fait qu'afficher le message "ping" dans la sortie standard (j'ai utilisé **object** au lieu de *class* tout simplement pour avoir un singelton).

J'ai aussi défini une méthode *doPeriodically* qui prend un *Action* et un entier comme paramètres, et qui répète l'exécution de l'action indéfiniment tout en "dormant" à chaque itération.

Voici à titre de comparaison la version Java de ceci:

```
interface Action {
  void doSomething();
}

class PingAction implements Action {
  @Override
  public void doSomething() {
    System.out.println("ping");
  }
}

public class TestMethods {
  private static void doPeriodically(Action action, int
period) {
    while(true) {
      action.doSomething();
      try{
        Thread.sleep(period);
      } catch (InterruptedException e) {
        return;
      }
    }
  }

  public static void main(String[] args) {
    doPeriodically(new PingAction(), 1000);
  }
}
```

2.2. La version fonctionnelle

En relisant le code précédent, on constate que malgré le fait qu'on a seulement besoin d'une méthode à exécuter périodiquement, on est obligé de construire plein de plomberie sans réel apport pour pouvoir y arriver, i.e. définir une interface, faire implémenter cette interface par une classe, et passage d'une instance de l'implémentation à la méthode d'exécution.

Or, ceci n'est aucunement nécessaire dans un langage fonctionnel comme Scala, où une méthode est un type de données comme les autres, c'est à dire que l'on peut écrire une méthode qui prend une méthode comme paramètre.

La définition du type d'une méthode en Scala se présente comme suit :

```
([paramètres]) => type-de-retour
```

Par exemple, une fonction mathématique comme le sinus ou le consinus admet le type suivant:

```
(Float) => Float
```

C'est à dire que c'est une fonction qui prend un *float* (réel) en paramètre et retourne aussi un réel.

Pour revenir à notre exemple, la méthode *doPeriodically* a besoin d'une méthode qui ne prend aucun paramètre et qui ne retourne rien, ce qui correspond au type :

```
() => Unit
```

Unit en Scala correspond au *void* de Java. Voici donc une nouvelle version du programme précédent utilisant ces nouvelles données :

```
object ScalaReMethods {
  def doPeriodically(action : () => Unit, period : Int)
= {
    while(true){
      action()
      Thread.sleep(period);
    }
  }

  def pong() = {
    println("pong")
  }

  def main(args : Array[String]) : Unit = {
    doPeriodically(pong, 1000)
  }
}
```

En gros, plus d'interface à implémenter ou d'objets à passer : on passe directement une méthode telle quelle à une autre fonction (via son nom), et celle-ci peut dès lors l'invoquer (toujours via son nom).

3. Concrètement

En fait, la notation "`(params) => type-du-résultat`" n'est que du sucre syntaxique fourni par Scala pour simplifier l'utilisation des fonctions.
Concrètement, les fonctions sont représentées par un ensemble de traits *FunctionN* où *N* dénote le nombre de paramètres qu'elle prend.

Par exemple, le type de fonction ne prenant pas de paramètres est représenté par *Function0[+R]*, où le *R* dénote le type de retour, tandis qu'une fonction prenant un seul paramètre est représentée par *Function1[-T1, +R]*, où *T1* est le type du paramètre et *R* le type de retour.

Dans tous les cas, ces traits définissent une méthode abstraite *apply* qui dans le cas de *Function1* par exemple est définie comme suit:

```
apply (v1 : T1) : R
```

4. Higher order functions

En utilisant ce qu'on vient de voir, il devient possible de coder des trucs vraiment intéressant avec les fonctions, genre une méthode qui combine deux fonctions en une nouvelle fonction par exemple.
Ce type de fonctions s'appelle fonctions d'ordre supérieur, ou "*Higer Order Function*" (Lien1).
A titre d'exemple, voici une implémentation en Scala de la fonction "rond", dénoté" par "o" en Mathématiques.
Petit rappel : `f o g (x) = f(g(x))`

```
type MathFunc = (Float) => Float

def rond(f : MathFunc, g : MathFunc) : MathFunc = {
  new Function1[Float, Float](){
    def apply(x : Float) = f(g(x))
  }
}

def main(args : Array[String]) : Unit = {
  val f : MathFunc = (x : Float) => 2 * x
  val g : MathFunc = (x : Float) => -1 * x
  val fog = rond(f, g)
  println(fog(5))
}
```

Je commence par déclarer un nouveau type *MathFunc* qui est `(Float) => Float`, où encore une fonction prenant un *float* comme paramètre et retournant un *float* (C'est strictement

optionnel, et uniquement dans le but de simplifier le code).

Je code ensuite la fonction rond qui prend deux *MathFunc* f et g en paramètre et retourne un *MathFunc*, i.e une nouvelle fonction de type *MathFunc* et qui devrait correspondre) f o g.
Comme résultat, cette méthode retourne une nouvelle instance de *Function1[Float, Float]* dont la méthode *apply* retourne *f(g(x))*.

Dans la méthode main, je montre un exemple d'utilisation de cette fonction en combinant deux fonctions f et g que j'ai défini.

Remarque : Veuillez notez que la composition de fonctions que je viens de montrer est déjà implémentée dans Scala par les traits *Function**, via la méthode compose.

Maintenant, à titre d'exemple (que je ne vais pas expliquer), et juste pour montrer la puissance de Scala et sa flexibilité, voici un bout de code qui permet de remplacer le "*rond(f, g)*" de l'exemple précédent par la notation plus naturelle "*f o g*" (La ligne surlignée en bleu clair):

```
type MathFunc = Fonction1[Float, Float]

class Roundable(f : MathFunc) {
  def o(g : MathFunc) = new MathFunc() {
    def apply(x : Float) = f.apply(g(x))
  }
}

implicit def toRoundable(f : MathFunc) = new Roudable(f)

def main(args : Array[String]) : Unit = {
  val f : MathFunc = 2 * _
  val g : MathFunc = -1 * _
  val fog = f o g
  println(fog(5))
}
```

5. Currying

Cette section a été corrigée suite à une remarque de SpaceGuid. Merci à lui

Le currying (Lien2) est une technique qui consiste à fixer la valeur d'un des paramètres d'une fonction pour en créer une nouvelle.
Par exemple, étant donné une fonction *mul* qui prend 2 paramètres x et y et qui retourne x*y (très intéressante et puissante comme fonction :)), on peut en dériver une fonction double qui prend un seul paramètre x et qui retourne son double (2*x) en fixant la valeur de y à "2".

Le currying est très simple et intuitif à mettre en place avec Scala. Jugez vous en par vous même à travers cet exemple :

```
object Currying {
  def mul(x : Int)(y: Int) : Int = x*y

  def main(args : Array[String]) : Unit = {
    val l = List(1, 5, 3)
    println(l.map(mul(2)))  //affiche List(2, 10, 6)
  }
}
```

En gros, j'ai fait le currying de la fonction mul(x)(y) en fixant la valeur du paramètre x à 2 (via mul(2)), ce qui donne naissance à une nouvelle fonction qui ne prend plus qu'un seul paramètre et qui le multiplie par 2.

6. Exemple: Collections Scala

Les collections de Scala utilisent intensivement l'aspect fonctionnel du langage, ce qui permet de réaliser des traitements complexes sur une liste d'une façon intuitive et fluide.

Je vais commencer par montrer quelques exemple simples, pour finir avec d'autres plus complexes.

6.1. Exemples simples

```
Object ScalaCollections {
  def main(args : Array[String]) : Unit = {
    val list = List(5, -1, 2, 27, 12, -3, 0, -6)

    //affiche l'indice du premier "0"
    println(list.findIndexOf((x) => x==0)) //affiche 6

    //affiche si oui ou non la liste contient 82
    println(list.exists((x) => x==82)) //affiche false

    //invoque println(x) pour chaque élément x de la
list
    list.foreach((x) => println(x)) //affiche toute la
liste

    //affiche éléments positifs de la liste
    println(list.filter((x) => x > 0))

    //affiche la somme des éléments de la liste
    println(list.reduceLeft((x, y) => x + y))
//affiche 36
  }
}
```

Les commentaires expliquent ce que fait chaque méthode.
Notez que j'ai usé du *type-inference* du compilateur Scala pour pouvoir écrire

```
(x, y) => x + y
```

par exemple en omettant les types des arguments :

```
(x : Int, y : Int) => x + y
```

6.2. Ne joues pas au plus "Perl" avec Scala !

Maintenant, je vais présenter les même exemples de la section précédente, mais en introduisant la "Perl touch", i.e. quelques hiéroglyphes magiques qui réduisent considérablement la taille de code (mais au risque de réduire autant la lisibilité du code).
L'hiéroglyphe magique en question dans Scala est le "_", qui représente le joker (% dans le SQL, ? dans les expressions régulières, * dans les noms de fichiers, etc.).
Il a plusieurs connotations dans Scala et ce selon où il est utilisé.
Dans le cas des *closures*, il permet de représenter un paramètre d'une fonction.
Par exemple, "`(x) => 2 * x`" peut être ré-écrit en "`2 * _`", et le compilateur Scala est assez malin pour savoir ce qui va dans le slot "_". De même, le compilateur peut se débrouiller avec un truc comme "`_ + _`" qui correspond à "`(x, y)=> x + y`" peut devenir .
Ainsi, le code précédant devient :

```
//affiche l'indice du premier "0"
println(list.findIndexOf(_==0)) //affiche 6

//affiche si oui ou non la liste contient 82
println(list.exists(_==82)) //affiche "false"

//invoque println(x) pour chaque élément x de la liste
list.foreach(println) //affiche toute la liste

//affiche éléments positifs de la liste
println(list.filter(_>0))

//affiche la somme des éléments de la liste
```

```
println(list.reduceLeft(_ + _))
```

6.3. Exemple plus complexe

```
//affiche la somme des carrés pairs des éléments de la liste
println(list.map(_ ^ 2).filter(_ % 2 == 0).reduceLeft(_+_))
```

No comment ! en fait si, un commentaire : dans Scala, ça bloque pas, c'est très fluide :)

7. Conclusion

Voilà, ainsi s'entame le troisième volet de l'introduction à Scala qui a servi, je l'espère, à présenter l'aspect fonctionnel de Scala via le.
Dans le prochain volet, ce serait le tour des *traits*, une sorte d'interfaces Java aux stéroïdes.

Retrouvez l'article de Jawher Moussa en ligne : Lien3

Les derniers tutoriels et articles

XMLBeans : mapping XML - JavaBeans

Le but de ce tutoriel est de présenter XMLBeans, une librairie de la fondation Apache.

1. Présentation

XMLBeans est un outil qui permet de faire du databinding Java/XML.
Il a été initié en 2003 par David Bau, de BEA, puis est devenu open source puisque cédé à la fondation Apache.
On peut télécharger la librairie sur http://xmlbeans.apache.org/ (Lien4)
La version actuelle est 2.3.0.
Elle offre la possibilité de mapper un document XML en objets Javabeans.
Au lieu de parcourir un arbre DOM ou de faire de l'évènementiel avec SAX par exemple, on passe par des getters et setters.
On peut générer des fichiers XML à partir de classes Java, ou inversement, générer des objets Java (POJOs) à partir de fichiers XML.

Les outils utilisés dans cet article sont :

- Eclipse 3.4 Ganymede
- Java 1.5
- Altova XMLSpy version 2008
- XMLBeans 2.3.0
- Saxon 8.8 (fichier saxonb8-8j.zip), une implémentation open-source de XSLT 2.0 et XPath 2.0, et XQuery 1.0

2. Schéma XML et DTD

Il est nécessaire de faire quelques rappels rapides.

- Une DTD (Document Type Definition) : une DTD permet de vérifier qu'un document XML est **valide**. Il est valide s'il est conforme aux règles sémantiques définies dans la DTD.
- Un schéma XML : aussi appelé XSD (XML Schema Definition), il est une alternative aux DTDs.

2.1. Validation via un Schéma XML

Vehicule1.xml :

```
<?xml version = "1.0" encoding = "UTF-8"?>
 <vehicules  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation = "vehicule1.xsd"  >
        <nom>moto</nom>
        <nombre>24</nombre>
        <couleur>verte</couleur>
 </vehicules>
```

A l'aide de XMLSpy (ou tout simplement d'un browser), on peut valider le fichier XML, par rapport au schéma XML suivant.

Vehicule1.xsd :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="vehicules">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="nom" type="xsd:string" maxOccurs="unbounded"/>
        <xsd:element name="nombre" type="xsd:integer" maxOccurs="unbounded"/>
        <xsd:element name="couleur" type="xsd:string" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

2.2. Validation via une DTD

Vehicule2.xml :

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE vehicules SYSTEM "vehicule2.dtd">
 <vehicules>
        <nom>moto</nom>
        <nombre>24</nombre>
        <couleur>verte</couleur>
 </vehicules>
```

Vehicule2.dtd :

```
<!ELEMENT vehicules (nom, nombre, couleur)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT couleur (#PCDATA)>
```

Les schémas XML présentent plusieurs avantages par rapport aux DTDs :
- une syntaxe XML
- possibilité de spécifier les types des données
- espace de nommage (namespace), ce qui implique que les règles pour définir certains éléments seront à un certain endroit tandis que les règles pour valider les règles pour définir d'autres éléments seront à un autre endroit.

3. Exemple

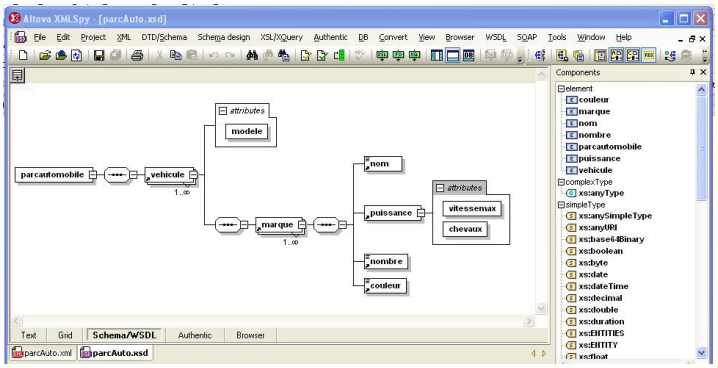
Pour ce tutoriel, je vais prendre comme exemple de base le cas d'un parc automobile qu'on peut découper de la façon suivante :

parcAuto.xml :

```
<?xml version="1.0" encoding="UTF-8"?>
<parcautomobile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="D:\WORK\XML\parcAuto.xsd ">
  <vehicule modele="coupe">
    <marque>
      <nom>Renault</nom>
      <puissance chevaux="8" vitessemax="280"/>
```

```
      <nombre>18</nombre>
      <couleur>noir</couleur>
    </marque>
    <marque>
      <nom>Peugeot</nom>
      <puissance chevaux="11" vitessemax="320"/>
      <nombre>10</nombre>
      <couleur>rouge</couleur>
    </marque>
  </vehicule>
  <vehicule modele="berline">
    <marque>
      <nom>Mercedes</nom>
      <puissance chevaux="10" vitessemax="330"/>
      <nombre>9</nombre>
      <couleur>gris</couleur>
    </marque>
    <marque>
      <nom>BMW</nom>
      <puissance chevaux="12" vitessemax="300"/>
      <nombre>7</nombre>
      <couleur>bleu</couleur>
    </marque>
  </vehicule>
</parcautomobile>
```

On peut automatiquement créer le Schéma XML grâce à XMLSpy :



parcAuto.xsd :

```
<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema generated by XMLSpy v2008 rel. 2 sp1
(http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="vehicule">
    <xs:complexType>
      <xs:sequence>
       <xs:element ref="marque" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="modele" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="berline"/>
            <xs:enumeration value="coupe"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
  <xs:element name="puissance">
    <xs:complexType>
      <xs:attribute name="vitessemax" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:short">
            <xs:enumeration value="280"/>
            <xs:enumeration value="300"/>
            <xs:enumeration value="320"/>
            <xs:enumeration value="330"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="chevaux" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:byte">
            <xs:enumeration value="10"/>
            <xs:enumeration value="11"/>
            <xs:enumeration value="12"/>
            <xs:enumeration value="8"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
  <xs:element name="parcautomobile">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="vehicule"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="nombre">
    <xs:simpleType>
      <xs:restriction base="xs:byte">
        <xs:enumeration value="10"/>
        <xs:enumeration value="18"/>
        <xs:enumeration value="7"/>
        <xs:enumeration value="9"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="nom">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="BMW"/>
        <xs:enumeration value="Mercedes"/>
        <xs:enumeration value="Peugeot"/>
        <xs:enumeration value="Renault"/>
      </xs:restriction>
  </xs:simpleType>
  </xs:element>
  <xs:element name="marque">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="nom"/>
        <xs:element ref="puissance"/>
        <xs:element ref="nombre"/>
        <xs:element ref="couleur"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="couleur">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="bleu"/>
        <xs:enumeration value="gris"/>
        <xs:enumeration value="noir"/>
        <xs:enumeration value="rouge"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
</xs:schema>
```

4. Génération des classes

Pour utiliser XMLBeans, il faut définir la variable d'environnement XMLBEANS_HOME, elle pointe vers le répertoire d'installation de XMLBeans.
On va créer les classes Java à partir du Schéma XML.

Pour cela, il y a 2 façons de faire. Soit on passe par l'utilitaire SCOMP, soit on utilise une tâche Ant xmlbean.

4.1. L'utilitaire SCOMP (Schema Compiler)

Dans une fenêtre DOS, on tape la commande **scomp** :

Après avoir ajouter XMLBEANS_HOME/bin dans le PATH (set PATH=%PATH%;%XMLBEANS_HOME%\bin\), on peut taper la commande suivante :

>scomp -src src -javasource 1.5 –out parcAuto.xsd

```
D:\tutorielXMLBeans>scomp -src src -javasource 1.5 -out parcAuto.xsd
Time to build schema type system: 1.25 seconds
Time to generate code: 0.735 seconds
Time to compile code: 3.078 seconds
Compiled types to: xmltypes.jar
```

Note: pour éviter d'avoir un java.io.IOException, il faut modifier le script scomp.cmd en changeant la ligne suivante :
java -classpath "%cp%" org.apache.xmlbeans.impl.tool.SchemaCompiler %*
par
%JAVA_HOME%/bin/java -classpath "%cp%" org.apache.xmlbeans.impl.tool.SchemaCompiler %*

On obtient la hiérachie suivante :

```
D:\tutorielXMLBeans\
D:\tutorielXMLBeans\parcAuto.xsd
D:\tutorielXMLBeans\xmltypes.jar  (contient les classes compilées et des fichiers XSB nécessaires à XMLBeans)
D:\tutorielXMLBeans\src (contient les interfaces et classes générées)
D:\tutorielXMLBeans\src\noNamespace (pas de namespace dans le Schema)
D:\tutorielXMLBeans\src\noNamespace\CouleurDocument.java
D:\tutorielXMLBeans\src\noNamespace\MarqueDocument.java (type complexe Marque)
D:\tutorielXMLBeans\src\noNamespace\NombreDocument.java
D:\tutorielXMLBeans\src\noNamespace\NomDocument.java
D:\tutorielXMLBeans\src\noNamespace\ParcautomobileDocument.java (element racine Parcautomobile)
D:\tutorielXMLBeans\src\noNamespace\PuissanceDocument.java
D:\tutorielXMLBeans\src\noNamespace\VehiculeDocument.java  (type complexe Vehicule)
D:\tutorielXMLBeans\src\noNamespace\impl
D:\tutorielXMLBeans\src\noNamespace\impl\CouleurDocumentImpl.java
D:\tutorielXMLBeans\src\noNamespace\impl\MarqueDocumentImpl.java
D:\tutorielXMLBeans\src\noNamespace\impl\NombreDocumentImpl.java
D:\tutorielXMLBeans\src\noNamespace\impl\NomDocumentImpl.java
D:\tutorielXMLBeans\src\noNamespace\impl\ParcautomobileDocumentImpl.java
D:\tutorielXMLBeans\src\noNamespace\impl\PuissanceDocumentImpl.java
D:\tutorielXMLBeans\src\noNamespace\impl\VehiculeDocumentImpl.java
```

Il faut inclure le jar xmltypes.jar dans le CLASSPATH.

4.2. Tâche Ant xmlbean

Une tâche ANT existe pour compiler un fichier XSD en classe XMLBeans. Elle permet d'obtenir un fichier JAR mais aussi le code source.
build.xml :

```
<?xml version="1.0" encoding="UTF-8" ?>
<project name="Tutoriel" default="xmlbean" basedir=".">

    <property name="xmlbeans.home" value="." />
    <property name="schema" value="schema/parcAuto.xsd" />
    <property name="dest.jar" value="dest/tuto.jar" />

    <path id="xmlbeans.path">
      <fileset dir="${xmlbeans.home}/lib" includes="*.jar"/>
    </path>

    <taskdef name="xmlbean"
             classname="org.apache.xmlbeans.impl.tool.XMLBean"
             classpathref="xmlbeans.path"
    />

    <target name="xmlbean">
       <xmlbean schema="${schema}"
                srcgendir="src"
                destfile="${dest.jar}"
                classpathref="xmlbeans.path" />
    </target>
</project>
```

Retrouvez l'article de Celinio Fernandes en ligne : Lien5

FAQ Eclipse

Où trouver Eclipse en français ?

Par défaut, seule la langue anglaise est utilisée dans Eclipse. Pour avoir le français, il faut téléchargez et installer un plug-in depuis l'update-site de **Babel** dont voici l'adresse :
http://download.eclipse.org/technology/babel/update-site/ (Lien6)

Les raccourcis claviers indispensables

Dans le désordre, quelques raccourcis claviers indispensables sous Eclipse (clavier configuration standard, et pas en configuration emacs), et passant bien sur les ctr + v, ctrl + s et autres que tout le monde connait

- **Ctrl + espace** : l'auto-complétion : si vous devez n'en connaitre qu'un seul, que ce soit celui-la
- **Ctrl + Shift + 0** : Organize Imports
- **Ctrl + Shift + R** : Recherche d'un fichier dans le workspace
- **Ctrl + O** : affichage des attributs et methodes de la classe courante
- **Ctrl + O une deuxieme fois** : ajoute à l'affichage les attributs et methodes héritées
- **Ctrl + T** : affiche l'arborescence d'héritage de la classe courante
- **Alt + Shift + J** : génére un template de javadoc pour une classe une méhode ou un attribut en fonction de la séléction
- **Ctrl + Shift + F** : mise en forme du code (vous pouvez surligner une zone de code pour restreindre le formatage)
- **Ctrl + Shift + I** : indentation du code (vous pouvez

surligner une zone de code pour restreindre l'indentation)
- **Ctrl + D** : efface la ligne courante
- **Alt + Shift + R** : pour lancer le "Rename Refactor", pour renommer une fonction ou une variable par exemple.
- **Ctrl + Shift + C** : pour commenter / decommenter les lignes sélectionnées
- **Crtl + Shift + P** : Pour se deplacer d'une accolade à l'autre

Comment définir ses propres templates ?

Les templates sont des bouts de code accessibles via des raccourcis typographiques.

Un exemple : tapez **sysout** quelque part dans votre code, et tapez le raccourci clavier de l'auto-complétion **Ctrl + Espace**, normalement vous devriez avoir un menu déroulant qui s'affiche sous votre curseur avec une entrée "sysout - print to standard out" (l'icône qui est à sa gauche est celui des "source code templates"). Sélectionnez cette entrée et tapez Entrée. Et voilà ! Le code "System.out.println();" a été inséré !

Remarquez au passage, la deuxième entrée du menu, elle propose de générer le squelette d'une fonction qui aura pour nom sysout. Cette génération est possible pour n'importe quel mot frappé. Si vous me suivez, vous aurez compris qu'en tapant un nom de fonction que vous voulez créer (par exemple "maFonction"), puis Ctrl + Espace, un squelette sera automatiquement généré (automatiquement à une condition : que le mot tapé ne correspondent pas à un raccourci typographique, comme "sysout") en prenant en compte vos préférences de génération de squelettes de fonction. Allez, ne perdez pas de temps et allez voir les templates de base, ou ajoutez vos propres templates de code dans le panneau de préférences (Menu Window->Preferences), section Java->Editor->Templates.

quelques exemples :

instanciation d'un objet :Tapez "new" quelque part. Tapez Ctrl + Espace, sélectionnez le template "new - create new Object", le code "type name = new type(arguments);" avec les champ à compléter encadrés, et le premier (le type) sélectionné. Choisissons pour l'exemple de créer un Integer, tapons "Integer". Remarquez comment le troisième champ (l'appel du constructeur) se modifie simultanément . Appuyez sur Tab pour passer au champ suivant (le nom de la variable) et au dernier (le paramètre du constructeur). boucle d'iteration : tappez for et ctrl + espace

Quels sont les langages que peut gérer Eclipse en plus de java ?

Grace a son architecture modulaire, eclispe peut gérér grace a des plugins un grand nombre de langages. Cette liste n'est pas exhaustive, et tous les plugins ne sont pas de qualités equivalentes.

- C/C++ : plugin officiel : www.eclipse.org/cdt (Lien7)
- Cobol : plugin officiel : www.eclipse.org/cobol (Lien8)
- DLTK : Plugin officiel qui oofre le support de Tcl, Ruby et Python : http://www.eclipse.org/dltk/ (Lien9)
- Php : PhpEclipse www.phpeclipse.de (Lien10)
- Python : pydev (Lien11)
- Latex : Texlipse (Lien12)
- Perl : e-p-i-c.sourceforge.net/ (Lien13)
- C# : improve-technologies esharp (Lien14), Emonic (Lien15)
- Ruby : rubyeclipse (Lien16)

Comment utiliser Ant avec Eclipse ?

Eclipse contient un plugin pour Ant par défaut. Il suffit de créer un nouveau fichier dans le répertoire et de le nommer **build.xml** pour que Eclipse le reconnaisse.
L'ouverture de ce fichier entraine l'activation d'un éditeur spécifique qui connait la syntaxe et la complétion de code.
Pour l'exécution soit un **clic droit sur le fichier et Run** soit dans le menu **Run -> External Tools -> Run as -> Ant Build**.

Comment developper en J2ME sous Eclipse ?

Utilisez le plugin EclipseMe : eclipseme.org (Lien17)
Suivez bien les instructions d'installation sur le site.

Retrouvez ces Q/R sur la FAQ Eclipse : Lien18

PHP : Un point sur la certification Zend

Apparue en 2004, la certification Zend est l'unique certification qui existe au monde sur PHP. Actuellement uniquement sur PHP5, elle a la particularité d'être plutôt complexe.
Plus que PHP, elle sert à évaluer les aptitudes du candidat sur le web et la programmation dans sa globalité. POO (Lien19), Design Patterns (Lien20), reconnaissance de failles de sécurité, protocole HTTP (Lien21), gestion des flux (Lien22) de données, XML et services webs (Lien23), sont autant de questions qui sont posées.
Etant moi-même formateur à la certification chez Anaska, je vous propose dans cet article, de passer en revue les sujets abordés, mais aussi les questions types, les astuces et les pièges à éviter.

1. PHP, une certification ?

Aujourd'hui, tout CV informatique, et *a fortiori* web, comporte le terme 'PHP'. Autant apprendre les bases de PHP peut se révéler simple, autant en maîtriser les arcanes nécessite un travail important. C'est cette double facette qui rend PHP complexe. Comment évaluer le niveau entre deux développeurs, l'un débutant prétentieux se définissant comme d'un bon niveau et de l'autre un expert humble qui se définit également d'un bon niveau ?
Cette certification est assez difficile et s'adresse à un public maîtrisant bien PHP. Cet examen témoigne que le candidat possède de bonnes compétences techniques en PHP et sur les technologies Web en général.
Ainsi, il est clair que tout "Zend Certified Engineer" (personne certifiée) est apte à travailler en entreprise sur des projets PHP, quelle qu'en soit la complexité.
Attention cependant, cela ne veut pas dire que le développeur est bon en terme de méthodologie, la certification est une validation technique.

L'examen de la certification Zend a été crée par un groupe de 12 experts en PHP à travers le monde. Zend est une société commerciale qui vend et supporte des produits de qualité autour de PHP dans un milieu professionnel.
Ce sont donc naturellement eux qui "portent" la certification, mais ils n'en sont pas les créateurs techniques (ils le sont en partie seulement).
La problématique de certification a longtemps été évoquée au sein du PHPGroup, mais c'est un énorme travail qui s'est transformé en serpent de mer. L'éditeur Zend (qui contribue largement au projet PHP) a donc pris le lead sur le projet et mis en place un examen de certification. Ainsi l'examen a été écrit par une communauté, pour une communauté : esprit PHP intact et conservé ;-) . Il s'agit d'une norme au sein des développeurs PHP dans le monde, et d'un gage de pérénnité au sein d'une entreprise.
Parmi les experts créateurs de l'examen, nous pouvons citer Damien Seguy, Ilia Alshanetsky, Christian Wenz, Mike Naberezny, Chris Shiflett, Marco Tabini, Marcus Boerger, Martin Jansen, Andi Gutmans, Zeev Suraski...

D'abord PHP4, la certification porte aujourd'hui uniquement sur PHP5 (5.1 pour être exact). De plus, elle a été entre temps traduite en Français, et elle est donc disponible dans notre langue depuis 2007.

2. Pourquoi se certifier ?

La certification complète un CV. Elle est le juste milieu entre experience et compétences théoriques (compréhension du système d'informations).
Elle ne peut pas combler un manque, elle ne peut pas remplacer un diplôme, elle sert de complément à un CV déjà rempli d'experiences.
Cependant, la personne ayant franchi le pas de la certification montre déjà une certaine implication dans la communauté PHP. PHP est un langage mûr, professionnel et apte à faire ses preuves dans le monde de l'entreprise et même de l'application critique (Le Monde, FaceBook, Skyrock, Yahoo).
La certification entre en jeu lors du recrutement d'une compétence. Pour distinguer un "bidouilleur", d'un programmeur chevronné (excusez ce parallèle grossier), la certification est une merveille. Un expert certifié Zend est assuré d'avoir suffisament de compétences sur PHP et son éco-système pour faire face à une mission professionnelle.

Les questionnaires techniques d'embauche de Yahoo, sur un poste PHP, sont plus difficiles que la certification PHP :-)

Une certification apportant la preuve d'une aptitude dans un domaine spécifique, l'interêt de la posséder n'est plus à démontrer, si toutefois PHP est votre domaine de prédilection.
Car le niveau reste relativement élevé : la certification Zend PHP5 est la seule qui existe sur le marché, et elle est reconnue internationalement. En France, actuellement seules 150 personnes (environ) sont certifiées. Dans le monde : environ 1500 (la certification datant de 2004...).
Zend a d'ailleurs mis en ligne un annuaire, appelé pages jaunes des ingénieurs certifiés Zend (Lien28), dans lequel apparait chaque certifié (seule liste officielle), peu de temps après l'obtention du diplôme.

3. La certification Zend de plus près

A la date de cet article, ce qu'il faut savoir :

1. QCM de 70 questions - 90 minutes
2. Le coût d'un passage avoisinne 150 euros
3. Aucun document autorisé, aucun manuel, pas de connexion Internet
4. 1 ardoise et 1 crayon à disposition
5. Répondre faux ne pénalise pas plus que ne pas répondre
6. On sait d'office le nombre de réponses à cocher (il peut y en avoir plusieurs)

7. Certaines rares questions demandent d'écrire le nom d'une fonction
8. En fin d'examen on peut revoir toutes les questions (si le temps le permet)
9. Lors de la validation finale, on sait immédiatement si on l'a ou pas
10. On ne connait ni le barème, ni la pondération, ni les réponses

La liste des points abordés :

1. Les bases de PHP : (tableaux, chaînes, autre : tout)
2. XML : SimpleXML, Xpath, Dom, Sax, LibXml2
3. Web services : SOAP, REST, XMLRpc, JSON
4. Design applicatif, règles de programation, algorithmique
5. POO, SPL, Design-Patterns
6. Flux et programmation réseau
7. Securité : failles, protections
8. Différences entre PHP4 et 5
9. Bases de données, SQL : PDO, SQLite, MySQLi
10. Web : Protocole HTTP, données GET/POST, Cookies/Sessions

La configuration supposée :

1. PHP5.1 (avec PDO donc)
2. Register globals off
3. Safe Mode off
4. Pas de questions particulières relatives à l'OS
5. Display errors à On
6. Pas de questions particulière sur le php.ini

Ces questions ne sont pas issues de la certification (il est interdit de les divulguer), mais elles sont très ressemblantes :
Quelle fonction peut être utilisée pour savoir si un flux est bloquant ou non ? stream_get_bloking | stream_get_metadata | stream_is_bloking | stream_get_blocking_mode
La méthode _____ peut être utilisée sur noeud SimpleXML pour retourner un itérateur contenant la liste de tous les noeuds fils
Exemple de questions de la certification PHP : http://www.certificationphp.com/tests/evaluation (Lien29)

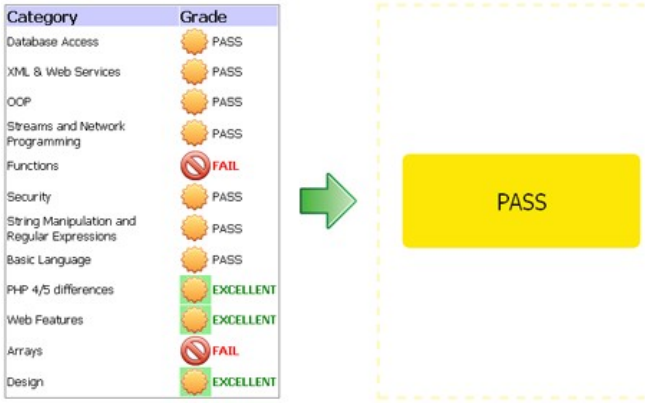
4. Les ressources autour de la certification

Vous n'êtes cependant pas seuls : il existe des livres pour étudier, et même des formations officielles permettant de se préparer :

- certificationphp.com (Lien25)
- Formation de préparation à la certification officielle par Anaska (Lien30) (5 jours).
- Zend PHP 5 Certification Study Guide (Lien31)
- Zend PHP Certification Pratice Test Book (Lien32)
- Zend PHP(4) Certification Study Guide (Lien33)

Tous ces livres sont en anglais. Concernant le livre sur PHP4 : je le conseille aussi, mais en ayant une bonne connaissance des différences PHP4/5. Par exemple le chapitre POO de PHP4 est à ignorer, sauf pour justement en étudier les changements par rapport à PHP5.
Les livres d'études PHP5 et PHP4 se complètent bien. Il reste un troisième ouvrage contenant des questions types, à ma connaissance celui-ci traite de PHP4.

Des tests blancs existent ! PHPArchitect (Lien34) vend des tests blancs, très ressemblants à l'examen réel.
Ces tests sont comme la vraie certification : aucune information de pondération, vous n'aurez aucunement les réponses aux questions à la fin du test. Vous pourrez en revanche apprécier un petit récapitulatif qui vous est présenté en fin d'examen blanc :



5. Le point de vue du formateur certifié

Avec mes stagiaires, nous faisons en général un test blanc à l'arrivée le premier jour, puis un le dernier jour avant de partir. Il y a bien entendu bonne progression dans les résultats observés.
La certification exige autant une expérience assise, que des compétences théoriques solides. On demande ainsi de connaitre une bonne partie des fonctions de PHP, mais aussi du web.
Les flux notamment, ou encore le protocole HTTP, les sessions et la sécurité. Quoi qu'il en soit, il faut vraiment bosser sur les fonctions str* et array* ; ainsi que sur le design applicatif.

Aussi, on vous demande souvent des noms de fonctions (parmis les options de réponse à cocher), et beaucoup de fonctions n'existent pas, ou pas bien orthographiées. Ainsi on peut proposer strsplit() (str_split()), ou encore headers_get() (headers_list()) ... Il faut réellement maitriser le langage.
On peut vous faire bien réfléchir sur les algorithmes tordus, avec des références de partout, des for() dans des while().

6. Conclusion et futur des certifications PHP

Cette certification tient son pari, d'un autre coté elle est unique (elle est la seule certification proposée sur le langage PHP). Il est fort peu probable qu'il existe une certification PHP6, car les changements PHP6/PHP5 seront très limités, Unicode et quelques autres petits changements : pas de quoi justifier une certification.
Cependant, il n'est pas impossible que Zend propose d'autres certifications... Je laisse un blanc volontaire ;-)

Retrouvez l'article de Julien Pauli en ligne : Lien35

Tutoriel : Installation d'une solution de paiement en ligne E-Transactions (SIPS-ATOS) sur un serveur Apache avec PHP

Ce tutoriel a pour but d'expliquer l'installation d'un système de paiement en ligne E-Transactions (SIPS-ATOS) sur un site e-commerce utilisant un serveur Linux avec Apache et PHP.

1. Généralités

1.1. Pré-requis

De bonnes connaissances de l'environnement Linux et de PHP sont nécessaires pour pouvoir installer une solution de monétisation sur un site web e-commerce.

PHP :

- Les sessions
- Les tableaux
- L'envoi d'email
- Lire et écrire dans un fichier

LINUX :
- Les chemins
- Les droits des fichiers

1.2. Introduction

Ce tutoriel devrait pouvoir s'appliquer probablement à toute solution de paiement en ligne qui utilise le système SIPS-ATOS, mais pour que l'exemple reste concret, nous allons nous pencher plus particulièrement sur le système E-Transactions du Crédit Agricole.

Nous allons étudier le cas de l'API pour Linux version 6 sur un serveur Linux OpenSuse (10.2) avec Apache (2.2.3) et PHP (5.2.5).

1.3. Présentation de l'API SIPS-ATOS

Dans ce tutorial, le package SIPS-ATOS nous a été délivré en pièce jointe d'un courrier électronique au format ***.zip**

Vous devez également avoir reçu votre certificat (Lien36) électronique de commerçant par email. Le certificat est empaqueté dans un fichier exécutable chiffré (Security Box). Vous recevrez par courrier postal le code qui permettra de déchiffrer votre certificat pour l'enregistrer sur votre disque dur.

Nous passerons ici sur toutes les formalités administratives relatives à la création de votre boutique en ligne auprès de votre établissement bancaire (constitution du dossier, présentation de documents administratifs, contrats et signatures) auxquelles vous devrez vous soumettre.

1.3.1. Contenu du package

Décompressez le fichier ZIP dans le répertoire de votre choix sur votre disque dur, puis ouvrez-le. Il devrait contenir :
- Un dossier **bin** : Il contient les exécutables **request** et **response**
- Un dossier **documentation** : Il contient toute la documentation nécessaire à l'installation ainsi qu"un guide du programmeur et un dictionnaire des données
- Un dossier **logo** : Il contient les logos de cartes bancaires
- Un dossier **param** : Il contient un certificat de test, ainsi qu"un fichier de paramètres du commerçant , un fichier de paramètres du fournisseur et le fichier **pathfile**
- Un dossier **sample** : Il contient des exemples de fichiers d'appel de requête de transaction et de réponse pour PHP et Perl
- Un dossier **test_template** : Il contient un Template d'exemple ainsi qu'un utilitaire de test test_template
- Un fichier de version

Dans ce tutoriel, nous laisserons de côtés tous les fichiers Perl (qui sont livrés avec).
Les principaux fichiers sont expliqués en détail plus loin.

1.3.2. Documentation

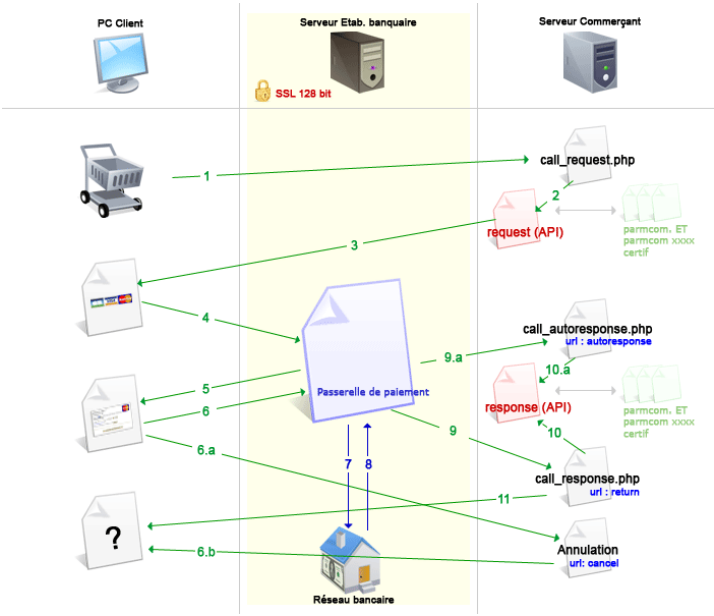
Ce tutoriel n'a pas pour but de se substituer à la documentation officielle qui vous a été fournie avec l'API (Lien37).

N'hésitez pas à lire les différents guides d'installation.
Lorsque vous serez un peu plus familier du système de paiement, vous vous pencherez sûrement sur le **Guide du programmeur** ainsi que le **Dictionnaire des données**. Ces documents vous permettront de passer des paramètres supplémentaires lors de la transaction et d'interpréter tous les messages de retour.
Le Dictionnaire des données vous sera également très utile : Chaque champ, chaque paramètre doit observer des règles de syntaxe très strictes.

1.4. Présentation du processus de paiement en ligne

Comme un schéma vaut mieux qu'un long discours, je vous laisse découvrir toute la chaîne du processus de paiement au travers du schéma ci-dessous.

1.4.1. Principe de fonctionnement d'une transaction en ligne (Schéma)



Shéma du déroulement d'une transaction en ligne

1. Le client a rempli le caddie et le valide pour procéder au paiement
2. Le fichier **call_request.php** est exécuté et interroge le binaire **request**
3. Affichage des moyens de paiement
4. Le client clique sur la carte bancaire. Les données de la transaction sont envoyées au serveur du fournisseur
5. Affichage du formulaire de saisie de carte bancaire
6. Le client saisit ses numéros de carte puis valide. (Si le client abandonne, il est redirigé vers la page d'annulation : 6a -> 6b)
7. Le serveur du fournisseur demande l'autorisation auprès d'une institution financière (réseau bancaire)
8. La réponse est traitée par le fournisseur
9. La requête est renvoyée vers le fichier de réponse automatique **call_autoresponse.php** et le fichier de réponse manuelle **call_response.php** (9 et 9.a)
10. Ces deux fichiers sont exécutés et interrogent le binaire **response** pour interpréter le résultat (10 et 10.a)
11. Le fichier de réponse manuelle **call_response.php** affiche la page de résultat (Succès ou échec)

2. Exemple d'installation pour une boutique en ligne avec un caddie simple (1 article)

2.1. Introduction

Dans cet exemple, considérons que notre boutique permette la vente de logiciels en ligne.
Le client, après s'être identifié, aura rempli un formulaire avec ses données personnelles nécessaires à la facturation, puis aura sélectionné un logiciel dans le catalogue.

Si vous n'en êtes pas encore au stade du caddie, je vous recommande alors vivement le tutoriel de Joris Crozier (Lien38)

sur la création d'un panier en PHP (Lien39)

Nous considérons que toutes ces données sont stockées dans une session que nous allons résumer ici comme ci-dessous :

Définition des variables de session

```
<?php

/* Informations du client */
$_SESSION['USER_ID']       = "1";
$_SESSION['USER_NOM']      = "Godin";
$_SESSION['USER_PRENOM']   = "Thierry";
$_SESSION['USER_COMPANY']  = "N1bus-Expériences";
$_SESSION['USER_ADRESSE']  = "1, rue Henri Desgranges";
$_SESSION['USER_VILLE']    = "La Rochelle";
$_SESSION['USER_ZIP']      = "17000";
$_SESSION['USER_PAYS']     = "France";
$_SESSION['USER_TEL']      = "33(0) 123 456 789";
$_SESSION['USER_EMAIL']    = "xxxx@xxxx.com";

/* Caddie */
$_SESSION['CADDIE_ID']          = "1";
$_SESSION['CADDIE_LOG_NOM']     = "PlanningFacile";
$_SESSION['CADDIE_LOG_VERSION'] = "1.1.0.0";
$_SESSION['CADDIE_AMOUNT']      = "3000"; //montant en
euro sans séparateur = 30,00 euros (C'est pas cher !)

?>
```

Nous considérons également que notre site web est installé dans le répertoire suivant sur le serveur :

```
/srv/www/htdocs/monsiteweb/
```

Et enfin, nous créons un répertoire qu'on appellera "xpay" pour y stocker les fichiers de configuration et les certificats :

```
/srv/www/htdocs/monsiteweb/xpay/
```

Nous n'avons pas besoin de connexion sécurisée pour notre boutique :
Toutes les données de la transaction seront chiffrées par le binaire **request** puis déchiffrées par le binaire **response**.
La transaction bancaire sera effectuée sur le serveur sécurisé (SSL (Lien40) 128 bit) de notre fournisseur.
Aucune information concernant la carte de crédit du client ne transite par notre serveur.
Pour une plus grande sécurité, vous pouvez évidemment utiliser le protocole HTTPS (Lien41) sur votre boutique, mais ceci vous obligera à acquérir et/ou installer un certificat SSL (Lien40) et probablement une adresse IP supplémentaire.

Nous utiliserons un serveur Linux en local pour les tests, configuré comme le serveur de production afin d'éviter les surprises.

2.2. Installation des fichiers binaires

Avant toute chose, vérifiez la configuration de PHP sur votre serveur avec la commande **phpinfo()**.

safe_mode : On

Si la directive safe_mode est à On, vous devez copier les exécutables **request** et **response** dans le répertoire défini par la directive **safe_mode_exec_dir**

safe_mode : Off

Si la directive safe_mode est à Off, vous pouvez copier les exécutables **request** et **response** dans le répertoire de votre choix à l'intérieur de votre site ou dans le répertoire **cgi-bin**.

Vous utiliserez les binaires correspondants à votre noyau Linux (2.4 ou 2.6)

Dans notre exemple :

- La directive **safe_mode** est à **On**
- Le répertoire **safe_mode_exec_dir** est : **/srv/www/htdocs/empty/**
- Le noyau : **2.6.18.8-0.9**

Nous allons donc copier les fichiers binaires **request_2.6.9_3.4.2** et **response_2.6.9_3.4.2** dans le répertoire : **/srv/www/htdocs/empty/**
CHMOD : 0755

2.3. Installation des fichiers de paramètres et des certificats

Afin de pouvoir s'y retrouver facilement, nous allons séparer les fichiers de l'API (Lien37) de paiement des fichiers du site, sauf en ce qui concerne les fichiers PHP d'appel et de réponse que nous mettrons avec les fichiers du site (à la racine).

Prenez soin de mettre des chemins en minuscule, sans accent et surtout sans espace.
Un espace dans le chemin d'un fichier provoquera une erreur

Nous allons copier les fichiers et dossiers suivants dans le répertoire : /srv/www/htdocs/monsiteweb/xpay/

- Fichier **pathfile**
- Fichier **parmcom.e-transactions** (Fichier de paramètres du fournisseur)
- Fichier **parmcom.013044876511111** (Fichier de paramètres pour les tests)
- Fichier **certif.fr.013044876511111** (Certificat pour les tests)
- Fichier **certif.fr.xxxxxxxxxxxxxxxxx** (Votre certificat de commerçant - avec votre numéro de commerçant délivré par votre fournisseur)
- Dossier **logo** (Contient les images des cartes bancaires)

Copiez ensuite le fichier **parmcom.013044876511111** dans le même répertoire et renommez-le en **parmcom.xxxxxxxxxxxxxxxxx** où **xxxxxxxxxxxxxxxxx** sera remplacé par le numéro que vous aura fourni votre fournisseur. (Ce sera probablement votre numéro de SIRET).
Ce nouveau fichier devra être modifié : vous mettrez les mêmes paramètres que dans le fichier **parmcom.013044876511111** après avoir effectué tous les tests.

Lorsque vous passerez en pré-production puis en production, il faudra utiliser votre certificat et votre fichier parmcom.....

2.3.1. Fichier pathfile

Le fichier **pathfile** (sans extension) contient les chemins vers les fichiers de paramètres, le certificat et le répertoire qui contient les images des cartes bancaires.

Contenu du fichier pathfile

```
##################################################
#
#       Pathfile
#
#       Liste fichiers parametres utilises par le
module de paiement
#
##################################################
#
#-------------------------------------------------
# Activation (YES) / Désactivation (NO) du mode DEBUG
#-------------------------------------------------
#
DEBUG!YES!
```

```
# ------------------------------------------------------
# Chemin vers le répertoire des logos depuis le web
alias
# ------------------------------------------------------
#
D_LOGO!xpay/logo/!
#
# ------------------------------------------------------
#  Fichiers parametres lies a l'api e-transactions
paiement
# ------------------------------------------------------
#
# fichier des  parametres e-transactions
#
F_DEFAULT!/srv/www/htdocs/monsiteweb/xpay/parmcom.e-
transactions!
#
# fichier parametre commercant
#
F_PARAM!/srv/www/htdocs/monsiteweb/xpay/parmcom!
#
# certificat du commercant
#
F_CERTIFICATE!/srv/www/htdocs/monsiteweb/xpay/certif!
#
# ------------------------------------------------------
#       end of file
# ------------------------------------------------------
```

Modifiez les chemins sans oublier le ! final.

Vous devez avoir au moins :

- Un chemin vers le répertoire des logos (chemin relatif)
- Un chemin vers le fichier du fournisseur : **parmcom.e-transactions!**
- Un chemin vers le fichier du commerçant : **parmcom!**
- Un chemin vers le fichier de certificat : **certif!**

Passez le paramètre **DEBUG** à **YES** pour les tests. Ceci affichera un message explicite au format HTML en cas d'erreur.
Nous repasserons ce paramètre à **NO** ensuite lors de la mise en pré-production

2.3.2. Certificats : fichiers certif.fr.xxxxxxxxxxx

Rien de particulier à dire sur les certificats (Lien42). Vous ne devez pas éditer ces fichiers.

2.3.3. Paramètres du commerçant : Fichier parmcom.xxxxxxxxxxxx

Le fichier **parmcom.xxxxxxxxxxxx** contient les Url de retour de la transaction.

Nous allons éditer le fichier de test **parmcom.013044876511111** puis nous recopierons ultérieurement les mêmes Url dans notre fichier **parmcom.xxxxxxxxxxxx**

Contenu du fichier parmcom du commerçant

```
########################################################
#
#       Fichier des parametres du commercant
#
#       Remarque :      Ce fichier parametre est sous
la responsabilite du
#                               commercant
#
########################################################


# URL de retour automatique de la reponse du paiement

AUTO_RESPONSE_URL!
http://www.monsite.com/call_autoresponse.php!

# URL de retour suite a paiement refuse

CANCEL_URL!http://www.monsite.com/call_response.php!

# URL de retour suite a paiement accepte

RETURN_URL!http://www.monsite.com/call_response.php!


# END OF FILE
```

Vous devez mettre des Url complètes

Modifiez les Url sans oublier le ! final.

Vous devez avoir au moins :

- Une URL vers le fichier de réponse automatique **call_autoresponse.php**
- Une URL vers le fichier de réponse manuelle **call_response.php**
- Une URL vers le fichier en cas d'annulation de la part du client. Nous utiliserons ici le même fichier **call_response.php**

Note:
Nous verrons plus loin que nous pourrons passer ces Url en paramètres lors de la requête directement dans le fichier **call_request.php**
Ceci peut être utile dans le cas où vous souhaiteriez passer d'autres paramètres dynamiques dans les Url. Dans ce cas, vous prendrez soin de commenter les lignes des Url dans ce fichier en ajoutant un # devant.

2.3.4. Paramètres du fournisseur : Fichier parmcom.e-transactions

Comme dans le fichier **parmcom** du commerçant, ces paramètres peuvent être passés lors de la requête directement dans le fichier **call_request.php**

Contenu du fichier parmcom du fournisseur

```
########################################################
#
#       Fichier des parametres E-TRANSACTIONS
#
#       Remarque :      Ce fichier parametre est sous
la responsabilite du CA
#
########################################################


# Mode d'affichage des blocs de paiment

BLOCK_ALIGN!center!

# Ordre d'affichage des blocs de paiement

BLOCK_ORDER!1,2,3,4,5,6,7,8!

# Mode de securite

CONDITION!SSL!

# Code devise  ( 978=EURO )

CURRENCY!978!
```

```
# flag d'edition des libelles des blocs de paiement

HEADER_FLAG!yes!

# Code langage de l'acheteur (fr=francais)

LANGUAGE!fr!

# Code pays du commercant

MERCHANT_COUNTRY!fr!

# Code langage du commercant

MERCHANT_LANGUAGE!fr!

# Liste des moyens de paiement acceptes

PAYMENT_MEANS!CB,1,VISA,1,MASTERCARD,1!

# Passage en une seule frame securisee au moment du
paiement

TARGET!_top!

# Nom du template de la page de paiement e-transactions

TEMPLATE!template_ca_fr!

# Couleur du text (noir)

TEXTCOLOR!000000!

# END OF FILE
```

Note :
La plupart des commentaires vous indiquent la nature des paramètres à configurer. Toutefois vous trouverez ci-dessous la description de certains paramètres particuliers.

2.3.4.1. Paramètre : PAYMENT_MEANS

C'est le paramètre qui affichera les images des différentes cartes de crédit avec le commentaire.

Il existe 3 blocs de commentaires :

N° de Block	Commentaire
1	Choisissez un moyen de paiement ci-dessous
2	Vous utilisez le formulaire sécurisé standard SSL, choisissez une carte ci-dessous :
4	Autres moyens de paiement :

Exemples :

Si vous renseignez ce paramètre avec les valeurs suivantes :

```
PAYMENT_MEANS
PAYMENT_MEANS!CB,1,VISA,1,MASTERCARD,1!
```

Les trois cartes de crédits seront affichées dans le bloc 1 sous le commentaire correspondant :

Si vous renseignez ce paramètre avec les valeurs suivantes :

```
PAYMENT_MEANS
PAYMENT_MEANS!MASTERCARD,2,CB,2,VISA,2!
```

Les trois cartes de crédits seront affichées dans le bloc 2 sous le commentaire correspondant :

2.3.4.2. Paramètre : BLOCK_ORDER

C'est le paramètre qui détermine l'ordre d'affichage des blocs.

Normalement la valeur par défaut **(1,2,3,4,5,6,7,8,9)** convient, mais si vous souhaitez afficher le bloc 2 avant le bloc 1, vous devrez alors renseigner ce champ avec les valeurs suivantes **(2,1,3,4,5,6,7,8,9)**

2.3.4.3. Paramètre : HEADER_FLAG

C'est le paramètre qui permet d'afficher ou masquer les commentaires

Si vous définissez ce paramètre à **no** , aucun commentaire ne sera affiché dans les blocs.

2.3.4.4. Paramètre : TEMPLATE

C'est le paramètre qui permet de définir le Template (Lien43) (design) qui sera utilisé pour le paiement sur le serveur du fournisseur.

Note :
Pour des raisons évidentes de sécurité, certaines parties de la page de saisie des codes de carte bancaire ne peuvent être modifiées (formulaire). En revanche, il est possible de modifier d'autres parties de la page, d'afficher votre logo, vos textes, vos couleurs.

Nous verrons plus loin comment créer un Template aux couleurs de votre site.
Pour le moment, nous laisserons le Template du fournisseur.

2.3.5. Autres fichiers et sous-dossiers

Le dossier logo :
Le dossier **logo** contient par défaut les images des 3 cartes bancaires (CB, VISA, MASTERCARD) et des cadenas SSL. Si vous devez utiliser d'autres cartes bancaires, assurez-vous d'y copier les images correspondantes.

Le fichier de logs :
Ce fichier n'existe pas par défaut, il vous faut donc créer un fichier texte vide que vous renommerez en log.txt (ou un autre nom si vous souhaitez). Il devra pouvoir être accessible en lecture/écriture par nos scripts PHP de paiement (**CHMOD 766**). Vous pouvez le mettre dans le répertoire de votre choix.

Nous utiliserons ce fichier pour enregistrer toutes les transactions (succès ou échec), ainsi , si aucun email de confirmation n'était délivré après une transaction (panne serveur, réseau, bug ...) , nous aurions tout de même une trace dans le fichier de logs.

De plus, en production nous y retrouverons les éventuels

messages d'erreurs de l'API.

2.4. Installation des fichiers PHP

Nous allons copier les 3 fichiers PHP à la racine du site dans : / srv/www/htdocs/monsiteweb/

- call_request.php
- call_autoresponse.php
- call_response.php

Astuce :
Pour que vous ne soyez pas déroutés par le code de votre site et celui du système de paiement, je vous recommande d'inclure ces fichiers dans vos pages. (**include()**).

En effet, ceci peut s'avérer utile si , pour une quelconque raison, vous étiez amenés à changer de système de paiement.
De plus, ces fichiers généreront juste les images des cartes bancaires pour le **call_request.php** et des messages de confirmation (succès ou échec) pour le **call_response.php**

Cela permettra de ne mettre que le code nécessaire au traitement du paiement sans s'occuper de la mise en forme.
Le code sera plus clair et donc plus facile à debugger ou à maintenir.

Dans ce cas, n'oubliez pas de modifier les Url de retour dans le fichier parmcom du commerçant.

2.5. Intégration du fichier d'appel (call_request.php)

2.5.1. Introduction

Ce fichier permet de préparer une transaction avant d'accéder au paiement proprement dit sur le serveur du fournisseur. C'est également ce fichier qui affichera les moyens de paiement (Cartes bancaires).

Pour que ce tutoriel soit le plus explicite possible, nous allons découper ce fichier en trois parties distinctes :

1. La récupération du caddie
2. La mise en place des paramètres de transaction
3. L'affichage des moyens de paiement

Vous retrouverez le fichier **call_request.php** complet dans la dernière section de ce chapitre.

2.5.2. Récupération du contenu du caddie

Comme nous l'avions défini au début de cet exemple, toutes les informations du client et le contenu du caddie sont déjà enregistrés dans des variables de session.
Nous allons donc récupérer ces variables pour pouvoir les transmettre lors de la transaction. Ceci nous permettra au retour de la transaction, d'envoyer un reçu de paiement détaillé au client par email et éventuellement de mettre notre base de données à jour (Si votre système de boutique en ligne utilise une base de données).

Une précaution, cependant :

Les paramètres de transaction obéissent à des règles syntaxiques très strictes. Par exemple, pour l'avoir déjà expérimenté à mes dépends, un espace dans la valeur d'un paramètre causera une erreur quasi incompréhensible car les données seront tronquées lors du traitement et l'API (Lien37) renverra un message d'erreur plus ou moins générique difficile à interpréter.

On retrouvera souvent ce type d'erreur lorsque les paramètres contiendront des caractères interdits ou certains caractères de ponctuation.
On peut résoudre le problème une fois pour toutes en utilisant la fonction **base64_encode()** sur les paramètres susceptibles de contenir ce type de valeurs. On utilisera **base64_decode()** ensuite lors du retour de la transaction pour récupérer ces valeurs dans leur intégralité.

Tout le contenu du caddie et les infos du client seront passés dans un seul paramètre lors de la transaction, le paramètre : **caddie**.
Nous allons donc mettre tout ça dans un tableau puis nous utiliserons la fonction **base64_encode()** pour nous assurer que les données ne causent pas d'erreur lors de l'appel de l'API (Lien37).

Préparation du caddie

```
<?php
// on construit le tableau du caddie en récupérant les
variables de la session
$TheCaddie = array();

// les infos du client
$TheCaddie[] =  $_SESSION['USER_ID'];
$TheCaddie[] =  $_SESSION['USER_NOM'];
$TheCaddie[] =  $_SESSION['USER_PRENOM'];
$TheCaddie[] =  $_SESSION['USER_COMPANY'];
$TheCaddie[] =  $_SESSION['USER_ADRESSE'];
$TheCaddie[] =  $_SESSION['USER_VILLE'];
$TheCaddie[] =  $_SESSION['USER_ZIP'];
$TheCaddie[] =  $_SESSION['USER_PAYS'];
$TheCaddie[] =  $_SESSION['USER_TEL'];
$TheCaddie[] =  $_SESSION['USER_EMAIL'];

//le contenu du caddie
$TheCaddie[] =  $_SESSION['CADDIE_ID'];
$TheCaddie[] =  $_SESSION['CADDIE_LOG_NOM'];
$TheCaddie[] =  $_SESSION['CADDIE_LOG_VERSION'];
$TheCaddie[] =  $_SESSION['CADDIE_AMOUNT'];

//on crée un numéro de commande pour le fun
$NumCmd      = "CMD-" . date ("YmdHis");
$TheCaddie[] =  $NumCmd ;

//pour envoyer le caddie à e-transaction sans probleme
//on serialize le tableau pour en faire 1 string et on
le base64_encode()
//car certains caractères sont interdits dans la valeur
du parm caddie

$xCaddie = base64_encode(serialize($TheCaddie));
?>
```

Retrouvez la suite de l'article de Thierry Godin en ligne : Lien44

Ce qu'il faut et ne pas faire en CSS - Partie 1 : La sélection CSS

CSS Globe démarre une série de petits articles appelés "Ce qu'il faut et ne pas faire en CSS". Ces séries ont pour but de pointer les mauvaises habitudes lorsqu'on utilise le CSS et en général les standards du Web. On tentera de répondre aux questions les plus fréquentes relatives au CSS. Cette première partie portera sur la sélection CSS.

1. Introduction

La sélection d'un élément est le processus de base pour appliquer un style en CSS. Si vous ne ciblez pas un certain élément vous ne pourrez pas le styliser, n'est-ce pas ? Cet article peut vous aider à corriger des erreurs que vous auriez pu commettre mais aussi à répondre aux questions relatives aux meilleures pratiques pour l'utilisation des sélecteurs en CSS.

Comme vous le savez, il y a plusieurs types de sélecteurs CSS (Lien45). Nous nous focaliserons sur leur utilisation pratique.

2. Classes & Id

```
<p class="signature">...</p>
<ul id="nav">...</ul>
```

N'utilisez pas des noms de classes trop détaillés comme "red", "blueDot" ou "roundedTop". Je n'utiliserai même pas de noms comme "left", "right" ou "large". Pourquoi ? Eh bien, imaginez que vous ayez conçu 2 colonnes sur votre site, une colonne nommée "left" (gauche) et l'autre nommée "right" (droite) et qu'à un moment vous deviez intervertir la position des colonnes. Votre client pourrait ne pas aimer votre thème bleu et vouloir le changer, eh bien vous êtes coincé avec un bouton de classe "blueSmall" (class="blueSmall")...

Le nom des classes devraient donc décrire le sens et le but du contenu mais pas son apparence. Ainsi au lieu d'utiliser un **div** avec **id="left"** aouter un **div** avec **class="main"** voire même **class="primary"**. Essayez le même nom de classe sur différents éléments, dans diverses situations. Les noms n'ont pas besoin d'être liés à un seul type d'élément. Vous pouvez utiliser **class="first"** sur le premier élément du menu ainsi que le premier paragraphe de votre contenu. Lorsqu'il faut choisir entre un nom de classe ou un id, déterminez si l'élément à styliser est vraiment unique. Dans ce cas, utilisez : **id**. S'il n'est pas unique utilisez les classes.

Cela vous évite d'utiliser plein d'id uniquement pour vos styles.

3. Les classes multiples

```
<button class="image submit">
```

Les noms de classes multiples sur un seul élément peuvent être une bonne façon d'optimiser votre css et d'éviter des noms de classes complexes pour le balisage.

J'utilise fréquemment une définition de classe, **class="image"**, qui possède un style général pour la technique de remplacement d'images. Puis j'ajoute des caractéristiques aux autres classes où je décris l'image à utiliser, je définis la largeur *etc.*

4. Les sélecteurs de types

```
a color:#ff0;
```

J'utilise toujours les sélecteurs de types au début de mes CSS, pour définir ou réinitialiser le style par défaut de certains éléments. Personnellement je ne crois pas à la puissance du sélecteur universel (à moins d'être utilisé en bas de hiérarchie). Si vous prévoyez d'utiliser des resets, utilisez ceux appropriés, c'est-à-dire le reset d'Eric Meyer (Lien46).

5. Les sélecteurs descendants

Utilisez-les ! Quand vous les maîtriserez vous trouverez que votre travail est beaucoup plus aisé contrairement à l'utilisation d'une centaine de noms de classes dans chaque partie de votre document. J'utilise fréquemment quelque chose comme **#main p a** pour rajouter un lien souligné que j'avais enlevé dans la réinitialisation.

6. Les combinaisons

```
#main h2.title span
```

Ce type de sélecteur vous permet de cibler avec précision un élément.

Le principe que j'utilise est le suivant : J'essaye de déterminer le conteneur le plus petit dans la hiérarchie qui possède des éléments non répétitifs. J'ajoute un nom de classe à ce conteneur et j'y indique les éléments cibles avec des sélecteurs de même type que ceux décrits plus haut. Si vous avez une balise **h2** que vous utilisez pour les titres, et que celle-ci possède seulement un span, il n'y a aucune utilité à ajouter un nom de classe à ce span. Vous pouvez la cibler avec un sélecteur de combinaison, comme vu précédemment.

7. Support multi-navigateur et multi-plateforme

Certains sélecteurs ne sont pas pris en charge par certains navigateurs, vous ne devez donc pas compter sur eux. C'est-à-dire, que vous ne pouvez compter que sur le pseudo-élément ":first-child " pour avoir les coins du haut arrondis, sachant qu'ils ne sont pas supportés par IE6. En vérité de nombreux utilisateurs utilisent toujours IE6 et il faut vivre avec ça.

8. Résumé

À faire

- Lorsque vous choisissez d'utiliser des noms de classes, utilisez des noms qui décrivent le sens et le but du contenu, et pas son apparence.
- Utilisez quelques noms de classes et utilisez-les sur divers éléments.
- Utilisez des sélecteurs de type pour (ré)initialiser le style par défaut. Si vous le souhaitez, utilisez des techniques de réinitialisation appropriées (mentionnées ci-dessus).

- Utilisez des noms de classes multiples sur un seul élément pour optimiser votre CSS.
- Utilisez des combinaisons de sélecteurs et ciblez précisément votre élément.

À ne pas faire

- Utiliser pas des noms de classes comme "red", "blueDot" *etc.*
- Utiliser pas des noms de classes compliqués ou des id comme "sideIntroSecondaryContentTop". Utilisez plutôt des sélecteurs descendants.
- Utiliser pas de sélecteur universel pour la réinitialisation du document.
- Utiliser pas des sélecteurs qui sont dépendant des navigateurs.
- Utiliser pas des noms de classes qui ne sont pas indispensables.

Retrouvez l'article de Alen Grakalic en ligne : Lien47

Créer un menu web 2.0 de style Digg-like

Cet article vous expliquera comment créer un menu Web 2.0 avec le style *Digg-like*.

1. Introduction

Ce tutoriel explique comment créer une barre de navigation telle que l'on voit souvent dans plusieurs *Digg-like* telles Digg.com. Le tout est bien simple, une petite poignée de CSS et de (X)HTML et le tour est joué. Le résultat devrait ressembler à ce qui suit :

2. Étape 1 - Le rendu (X)HTML

Créer une nouvelle page HTML et copier le code suivant entre vos balises **<body>** et **</body>** :

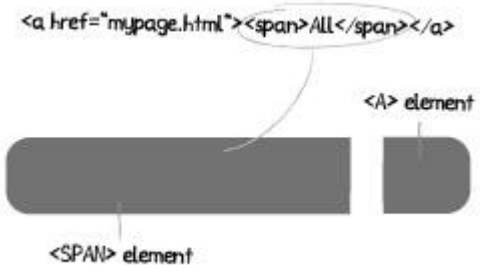
```
<div id="topbar">
    <a href="/toppage1.html"><span>All</span></a>
    <a href="/toppage2.html"
class="active"><span>News</span></a>
    <a href="/toppage3.html"><span>Video</span></a>
    <a href="/toppage4.html"><span>Images</span></a>
</div>
<div id="middlebar">
    <a
href="/midpage1.html"><span>Technology</span></a>
    <a href="/midpage2.html"><span>World</span></a>
    <a href="/midpage3.html"><span>Science</span></a>
    <a href="/midpage4.html"><span>Gaming</span></a>
</div>
```

Le second lien "**News**" est associé à la classe CSS «**active**». Qui représente la page en cours. Si vous utilisez les variables URL et PHP afin d'implémenter une barre de navigation dynamique, il vous sera facile d'implanter le tout facilement. Le code pour implanter ceci devrait ressembler à quelque chose du genre :

```
<?php if(isset($_GET['news'])){ echo 'class="active"';}
?>
```

3. Étape 2 - Implanter les coins arrondis à vos boutons

Avant de continuer, je vais vous expliquer comment implanter ce superbe effet CSS. Le tout est très simple.



...dans le code (X)HTML :

Images utilisées :

-
-
-
-

```
<a href="/toppage1.html"><span>All</span></a>
```

4. Étape 3 : CSS et #topbar

Créer un nouveau fichier CSS nommé **style.css** et copier le code suivant pour l'élément **#topbar**. N'oubliez surtout pas de lier votre page HTML à votre fichier CSS.

```
#topbar
{
    font-size:14px;
    color:#3b5d14;
    background:#b2d281;
    font-weight:bold;
    padding:6px;
    overflow:auto;
    height:1%;
    clear:both;
}

#topbar a
{
    color:#3b5d14;
    text-decoration:none;
    margin:0 10px;
    height:23px;
    line-height:23px;
    float:left;
    display:block;
}

a.active
{
    height:23px;
    line-height:23px;
    background:url(image/tb_a.png) right top no-repeat;
    padding-right:10px;
}

a.active span
{
    background:url(image/tb_span.png) left top no-
repeat;
    height:23px;
    display:block;
    padding-left:10px;
}
```

5. Étape 4 : CSS et #middlebar

Dans le même fichier CSS, ajouter les lignes suivantes pour l'élément **#middlebar**.

```
#middlebar
{
    font-size:11px;
    color:#3b5d14;
    background:#90b557;
    font-weight:bold;
    padding:6px;
    overflow:auto;
    height:1%;
    clear:both;
}

#middlebar a
{
    color:#3b5d14;
    text-decoration:none;
    margin:0 5px;
    padding-right:10px;
    height:23px;
    line-height:23px;
    display:block;
    float:left;
    background:url(image/mb_a.png) right top no-repeat;
}

#middlebar a span
{
    background:url(image/mb_span.png) left top no-repeat;
    height:23px;
    display:block;
    padding-left:10px;
}
```

Enregistrer le tout, et tester votre joli petit menu ! N'est-ce pas beau ? Peu importe, le tout est très classe et à la mode web 2.0. Si vous avez des questions, laisser un commentaire nous nous ferons un plaisir de vous répondre.

Retrouvez la suite de l'article de Dave Lizotte en ligne : Lien48

Présentation de F#

F# est un langage fonctionnel et objet de la famille ML, très proche du langage OCaml (Lien49) et en grande partie compatible avec celui-ci. Cependant, F# est un langage à part entière, qui est totalement intégré la plate-forme **.Net**. F# est développé par une équipe du Microsoft Research Center de Cambridge (UK).

1. Présentation

F# est un langage dérivé de *Caml* et conçu spécifiquement pour la plateforme *.NET*. C'est un langage fonctionnel (et incitant ce mode de programmation), entièrement orienté objet (même les entiers sont des objets et peuvent avoir des méthodes) et supportant la programmation impérative.

Il est dérivé de Caml : il est même possible d'écrire des programmes qui puissent être compilés à la fois par F# et par Caml. C'était par exemple le cas du compilateur F# lui-même, pendant longtemps. Malgré cette proximité, il s'agit bien de deux langages distincts. Certaines fonctionnalités de Caml ne sont pas acceptées par F# ; à l'inverse, F# possède de nombreux ajouts par rapport à Caml.

F# a été créé spécifiquement pour la plateforme .NET. Il bénéficie d'une bonne intégration (contrairement à certains langages qui ont été simplement portés pour .NET). Cela lui permet aussi d'accéder à une très large bibliothèque, de pouvoir s'utiliser avec l'*ASP*, avec *Silverlight*, avec *XNA*, etc.

C'est un vrai langage fonctionnel et il possède toutes les fonctionnalités que l'on attend dans ce paradigme. Il permet de créer ses structures de données de façon très concise et puissante. Il possède un typage statique et fort, et de l'inférence de types. Cela réduit considérablement le débogage et permet de faire du code sûr. Du code F# qui compile a toutes les chances de fonctionner.

F# est un langage très concis et très expressif. Sa verbosité est comparable à ce que l'on trouve dans Python ou Ruby. Il est très souple, peut utiliser du typage dynamique lorsque c'est nécessaire, peut utiliser l'évaluation paresseuse quand il le faut, possède un système de filtrage par motif (pattern matching) extensible, permet de manipuler des flux de manière poussée (via les compréhensions et un système de monades), etc.

Au final, F# est un langage innovant et possédant de nombreuses fonctionnalités uniques. Il est issu de la recherche de chez Microsoft. Il est encore jeune, mais il évolue très rapidement.

2. Mise en place

2.1. Installation

F# est un langage portable qui nécessite la plateforme .NET. .NET est installé par défaut sur certains Windows, il est aussi possible de le récupérer sur le site de Microsoft. De nombreuses personnes utilisent F# sous Linux, MacOS ou d'autres Unix. Pour cela, il est recommandé d'utiliser une version récente de Mono (Lien50).

Ce langage évoluant très rapidement, il est conseillé de récupérer la dernière version sur le site de Microsoft Research (Lien51) et de le mettre à jour régulièrement. La dernière version a de légers problèmes d'installation, voyez comment installer F# 1.9.4.17 sous Mono (Lien52).

2.2. Éditeur

Visual Studio est l'éditeur conseillé pour faire du F# sous Windows. Le plugin pour Visual Studio est systématiquement fourni avec le compilateur F#. Il est possible de télécharger une version gratuite de Visual Studio (Lien53), qui accepte le plugin F#. Ce plugin offre entre autres la coloration syntaxique, la vérification du code (syntaxe et typage) en temps réel, un mode interactif et la complétion contextuelle.

Il existe d'autres éditeurs pouvant être utilisés, aussi bien sous Windows que sous les autres plateformes. Si votre éditeur habituel ne possède pas encore de mode pour F#, regardez s'il en possède un pour OCaml, la syntaxe étant proche. Pour Emacs, il y a notamment un mode F# (Lien54), gérant la coloration du code, le mode interactif et l'indentation automatique (mais il manque encore de maturité).

2.3. Hello World

Pour tester votre installation, voici le Hello world.

```
System.Console.WriteLine "Hello world!"
```

Ceux qui connaissent .NET doivent reconnaitre la méthode utilisée. Les autres peuvent l'ignorer, car il y a une autre solution, plus courte.

```
printfn "Hello world!"
```

La fonction printf doit être familière pour ceux qui ont fait du C. Elle affiche à l'écran le texte donné en argument et supporte de nombreux formats pour afficher chaines de caractères, nombres et autres types. La fonction printfn correspond à un appel à printf, suivi d'un saut de ligne.

Retrouvez l'article de Laurent Le Brun en ligne : Lien55

F# : Types de base, expressions et déclarations

Dans cet article, vous retrouverez les fondements de F# : types de bases, expressions, déclarations de valeurs. Les fonctions seront abordées dans un article ultérieur.
Attention, n'oubliez surtout pas de consulter la dernière partie consacrée au mode *#light* de F# qui permet d'obtenir une syntaxe encore plus concise de son code, au prix d'une indentation rigoureuse.

1. Introduction

La grammaire de F# est relativement simple et consistante. Il n'y a que deux types d'éléments : les expressions et les déclarations. Quand une expression est évaluée, elle se simplifie en une valeur simple. Toutes les valeurs ont un type (et un seul).

Si vous avez accès au mode interactif de F# (fsi.exe), je vous conseille de l'essayer. Dans ce cours et les suivants, beaucoup d'exemples sont donnés. Je vous recommande de tester ces exemples par vous-même pour mieux comprendre. Dans le mode interactif, l'invite de commande est représentée par un chevron <. Il faut taper **;;** suivi d'entrée pour envoyer la commande à l'interpréteur. L'interpréteur évalue ensuite la commande. Si la commande est une expression, elle est simplifiée. La valeur de retour est affichée, ainsi que son type.

Pour se familiariser avec la syntaxe, voici des exemples d'expressions. En général, ça devrait assez simple à comprendre. Les commentaires en F# sont :

- // commente jusqu'à la fin de la ligne
- **(*** commente jusqu'au ***)** associé (ça peut être imbriqué)

Dans les exemples qui suivent, j'ai utilisé des valeurs de base et quelques opérateurs. On verra les fonctions dans un prochain cours.

2. Les Types de base

2.1. Les Entiers

```
> 42;;
val it : int = 42
```

Le type de l'expression "42" est donc *int* (entier). Sa valeur est "42".

```
> 5 + 5;;
val it : int = 10
```

Le type de l'expression "5 + 5" est donc *int*. Sa valeur est "10".

```
> 4 * (5 - 3);;
val it : int = 8
> (5 + 6) % 10;;          // % correspond au modulo.
val it : int = 1
```

2.2. Les Flottants

```
> 4.;;
val it : float = 4.0
> 4.0;;
val it : float = 4.0
> 4.5;;
val it : float = 4.5
> 4.2 + 5.3 * 4.1;;      // + fonctionne aussi sur les flottants
val it : float = 25.93
> 2. ** 8.;;             // ** est l'opérateur puissance.
val it : float = 256.0
```

2.3. Les Caractères

```
> 'a';;
val it : char = 'a'
> '\n';;                 // saut de ligne
val it : char = '\n'
> '\'';;
val it : char = '\''
> ''';;                  // raccourci syntaxique
val it : char = '\''
```

2.4. Les Booléens

```
> true;;
val it : bool = true
> false;;
val it : bool = false
> 4 = 5;;                // en C/C++ : ==
val it : bool = false
> 4 < 42;;
val it : bool = true
> 4 <> 42;;              // en C/C++ : !=
val it : bool = true
> 'a' < 'b';;
val it : bool = true
> true || false;;
val it : bool = true
> 3 < 4 && 'c' > 'a';;  // && a une faible priorité.
val it : bool = true
```

2.5. Les Chaînes de caractères

```
> "test";;
val it : string = "test"
> "Hello " + "world!";;            // concaténation
val it : string = "Hello world!"
> "test".[0];;                     // Accès à un caractère (ça commence à 0)
val it : char = 't'
> "test".[1];;
val it : char = 'e'
> "c:\\games\\";;
val it : string = "c:\\games\\"
> @"c:\game\";;          // chaine "verbatim" : les \ ne sont pas interprétés
val it : string = "c:\\games\\"
> "test".[0..2];;        // sous-chaine
val it : string = "tes"
> "test".[2..3];;
val it : string = "st"
> "test".[2..2];;
val it : string = "s"
```

2.6. Les Listes

Les listes sont toujours paramétrées par un type : *int list* correspond au type liste d'entiers. Il peut aussi être noté list<int>. Niveau implémentation, le type liste correspond à une liste chainée.

```
> [1; 4; 6; 10; 5];;     // liste litérale
val it : int list = [1; 4; 6; 10; 5]
> ["this"; "is"; "a"; "test"];;
val it : string list = ["this"; "is"; "a"; "test"]
> 2 :: [3; 4];;          // :: est l'opérateur de construction de listes
val it : int list = [2; 3; 4]
> 1 :: 4 :: [5];;        // l'opérateur :: est associatif à droite
val it : int list = [1; 4; 5]
> 1 :: 4 :: 5 :: [];;
val it : int list = [1; 4; 5]
> [4; 5] @ [10; 4];;     // @ est l'opérateur de concaténation de listes
val it : int list = [4; 5; 10; 4]
> [1..10];;              // range comprehension
val it : int list = [1; 2; 3; 4; 5; 6; 7; 8; 9; 10]
> [3..2..10];;           // pareil, avec incrément
val it : int list = [3; 5; 7; 9]
> [10 .. -1 .. 5];;
val it : int list = [10; 9; 8; 7; 6; 5]
```

2.7. Les Tableaux

De la même façon que les listes, ils sont paramétrés par un type. Les éléments d'un tableau sont stockés ensemble dans la mémoire, ce qui permet un accès direct.

```
> [| 3; 6; 10; 5 |];;
val it : int array = [|3; 6; 10; 5|]
> [| 4; 10; 5 |].[0];;
val it : int = 4
> [| 4; 10; 5 |].[1];;
val it : int = 10
> [| 4; 10; 5 |].[0..1];;       // sous-tableau
val it : int array = [|4; 10|]
> [| 'a' .. 'e' |];;
val it : char array = [|'a'; 'b'; 'c'; 'd'; 'e'|]
> [| 8 .. 12 |].[2];;
val it : int = 10
>  [| 80 .. 3 .. 100 |].[2..5];;
val it : int array = [|86; 89; 92; 95|]
```

2.8. Les Tuples

Un tuple permet de regrouper plusieurs valeurs. Ça peut être vu comme une structure anonyme, dont les champs sont anonymes. Voici quelques exemples, regardez bien le type des valeurs :

```
> 2, 3;;
val it : int * int = (2, 3)
> "test", 4;;
val it : string * int = ("test", 4)
> 2, 4, 10;;
val it : int * int * int = (2, 4, 10)
> [2; 3], 4, "test";;
val it : int list * int * string = ([2; 3], 4, "test")
```

Pour ce dernier exemple, le type indique que c'est un tuple composé d'une liste d'entiers, d'un entier et d'une chaine de caractères.

```
> [1, "this"; 2, "is"; 3, "a"; 4, "test"];;
val it : (int * string) list = [(1, "this"); (2, "is");
(3, "a"); (4, "test")]
```

Ceci est une liste de couples.

3. Les Expressions

3.1. Du typage fort

Les expressions suivantes sont mal typées et génèrent une erreur dès la compilation.

```
4 + 4.2 // on n'ajoute pas deux valeurs de types
différents

'a' + 1  // même remarque. Et il n'y a jamais de cast
implicite

[2; 4; "test"] // les éléments d'une liste doivent tous
avoir le même type

[| 4; true |]  // pareil pour les tableaux

4 = 4. // on ne compare pas deux types différents

(4, 3) = (4, 5, 2) // les tuples n'ont pas le même
type.

(3, 'a') = ('a', 3) // même problème : l'ordre dans un
tuple est important.
```

Pour convertir un entier en flottant, on utilise la fonction *float*. Pour la conversion inverse, c'est *int* (tronque la partie décimale).

3.2. Les Expressions conditionnelles

Les expressions conditionnelles ont la syntaxe suivante :

```
if <expr1> then <expr2> else <expr3>
```

expr1, expr2 et expr3 sont des expressions quelconques. Il faut toutefois que expr1 s'évalue en type bool, et que expr2 et expr3 aient le même type t. Cette expression renverra soit l'évaluation de expr2 (si expr1 vaut true), soit l'évaluation de expr3 (si expr1 vaut false).
Ainsi, le *if then else* est l'équivalent de l'opérateur ternaire du C. Il renvoie toujours une valeur. Et il n'y a pas d'équivalent au *if* du C.

```
> if true then 4 else 5;;
val it : int = 4
> if 4 > 5 then "test" else "foo";;
val it : string = "foo"
> "abcdefgh".[if 3 * 3 < 5 then 0 else 2];;
val it : char = 'c'
```

Partout où l'on peut mettre une expression, on peut mettre une condition.
Les deux expressions suivantes sont mal typées :

```
if 4 then 1 else 2              // 4 n'est pas un
booléen


if true then 4.5 else 2         // 4.5 et 2 n'ont pas le
même type
```

4. Les Déclarations

4.1. Les Déclarations locales

Pour associer un nom à une valeur, on utilise la construction *let..in*. Elle a cette syntaxe :

```
let <ident> = <expr1> in <expr2>
```

expr1 et expr2 sont deux expressions quelconques, de n'importe quels types. Pour l'identifiant, il faut suivre (en gros), cette expression rationnelle : *[a-zA-Z_][a-zA-Z_0-9']**
Ainsi, les identifiants suivants sont valides :

- toto
- Test
- g'
- _foo_bar42

Dans expr2, on peut utiliser l'identifiant. Il aura la même valeur que exp1. Voici quelques exemples de définitions locales :

```
> let x = 6 in 6 * 6;;
val it : int = 36
> let x = "a" in x + x + x;;
val it : string = "aaa"
```

Le bloc "let in" étant lui-même une expression, il est possible de les imbriquer.

```
> let x = 5 in let y = x + 1 in x + y;;
val it : int = 11
```

Partout, absolument partout, où l'on attend une expression, il est possible de définir localement une valeur. Par exemple :

```
> [| 1 .. let x = 3 in x * x |];;
val it : int array = [|1; 2; 3; 4; 5; 6; 7; 8; 9|]

> "test".[let x = 3 in x - 2];;
val it : char = 'e'
```

Avec la construction *let in*, on peut masquer une définition déjà existante. Ainsi, si on écrit *let x = x + 1*, le x de l'expression fait référence à un x déjà défini auparavant. Par exemple :

```
> let x = 2 in let x = x + 1 in x;;
val it : int = 3
```

Bien sûr, ce genre de construction est à éviter, mais c'est pour montrer comment est gérée la portée.

4.2. Les Déclarations globales

On souhaite parfois définir une valeur de façon globale. On utilise pour cela la construction *let <ident> = <expression>*. L'identifiant est alors visible dans la suite du programme.
Ainsi :

```
> let x = 4;;
val x : int
> x;;
val it : int = 4
> x + 1;;
val it : int = 5
```

Il est important de noter qu'une définition globale n'est pas une expression. On ne peut donc pas l'utiliser là où une expression est attendue.

```
> let a =
    let x = 5 in
    let y = 6 in
    x * y;;
val a : int
> a;;
val it : int = 30
```

5. Mode #light

Pour activer le mode *light*, il suffit de taper **#light** dans le mode interactif ou de mettre cette commande au début du fichier. Quand il est activé, la grammaire est légèrement modifiée : elle est allégée et se base sur l'indentation pour désambiguiser la syntaxe. Cela impose une certaine rigueur (même si les règles d'indentation sont relativement souples), mais permet d'avoir du code bien plus court et lisible. Ce mode light n'interdit pas d'utiliser le "in" si on le souhaite (ce qui est pratique lorsque l'on veut mettre une expression sur une ligne), ni l'utilisation de blocs explicites.
Les spécifications formelles de la syntaxe sont un peu complexes, mais elles sont plutôt intuitives. Par la suite, j'utiliserai toujours (sauf mention contraire) cette syntaxe light. Et je vous conseille fortement de l'utiliser systématiquement, d'autant plus qu'il sera mis par défaut dans une prochaine version.
Le dernier exemple donné devient :

```
> let a =
    let x = 5
    let y = 6
    x * y;;
val a : int
```

Comme vous pouvez le voir, tous les "in" peuvent être omis. Nous verrons les autres différences plus tard. Vous pouvez vous rendre compte que le code n'est pas ambigü : la valeur de a étant forcément une expression, les définitions de x et de y ne peuvent être que locales.
D'une manière générale, deux valeurs qui ont la même portée ont la même indentation. Pour l'indentation, vous pouvez utiliser le nombre d'espaces que vous souhaitez, mais soyez consistants. Et surtout, n'utilisez jamais de tabulation (configurez votre éditeur si nécessaire). Voici un exemple complet, qui compile, et qui reprend la plupart des choses vues jusqu'à maintenant (regardez bien l'indentation) :

```
#light

let a =
  let x = 10
  let y = 15
  x * y

let b = a % 5

let c =
  if b < 3 then
    "oui"
  else
    "non"

let d = c.[0..1]

printfn "a = %d, b = %d, c = %s, d = %s" a b c d
```

```
$ fsc test.fs
$ ./test.exe
a = 150, b = 0, c = oui, d = ou
$
```

Retrouvez l'article de Laurent Le Brun en ligne : Lien56

Tutoriel : Utiliser Silverlight 2 avec MySQL en C#

Cet article présente une solution pour utiliser Silverlight avec une base de données MySQL. On y verra également quelques utilisations des composants de Silverlight.

1. Introduction

Vous êtes un développeur C#, débutant avec Silverlight et vous souhaitez utiliser Silverlight avec une base de données MySQL ?
Alors ce tutoriel est pour vous.

A travers cet article, nous allons présenter une méthode pour accéder à une **base de données MySQL**, exécuter des requêtes de sélection, d'insertion, etc ...
Tout au long de ce cours, je vais utiliser Visual C# 2008.

Notez qu'au moment où j'écris cet article, Silverlight est encore en version 2 béta 2 et j'utilise certains hacks pour contourner des problèmes qui n'existeront sans doute plus dans la version finale de Silverlight.

2. Préambule

Dans ce tutoriel, je considère que vous avez quelques notions de Silverlight et les outils correctement installés.
Pour plus de renseignements sur l'installation des outils Silverlight, vous pouvez aller consulter l'introduction à Silverlight 2 (Lien57) de Benjamin Roux.

3. Le besoin : utiliser MySQL et PHP avec Silverlight

On voit un peu partout des utilisations de Silverlight avec Sql Serveur, notamment à travers l'utilisation de LINQ et de WCF. Je vais vous montrer comment on peut utiliser Silverlight avec une base de données MySQL par exemple.

L'intérêt est de pouvoir utiliser des applications Silverlight dans un environnement open source (LAMP par exemple) ou de les héberger chez votre fournisseur d'accès qui en général propose un serveur apache avec MySQL *(comme Free par exemple)*.

Comme on ne peut pas utiliser **Linq To Sql** avec autre chose

qu'Sql Server, l'idée est d'utiliser un script PHP qui accèdera à la base de données MySQL qui servira d'interface avec notre application **Silverlight**.
Ainsi, quand on aura besoin d'informations, notre script PHP ira lire dans la base de données et nous renverra les informations en format XML que l'on pourra exploiter avec **Linq To Xml**.
De même, quand on aura besoin de faire des insertions ou des mises à jour, on enverra des données à un script PHP qui s'occupera de communiquer avec la base de données MySQL.

4. Lire une table de la base MySQL

Imaginons que nous voulions faire une application qui affiche une *todolist*. Le premier besoin est d'être capable de lire dans une table (qui s'appellera todolist) avec un select. Cette table contient 2 champs :

- **id**, qui est un entier auto incrémenté
- **libelle**, qui est une chaîne de caractères *(varchar(500))*

Le but est de récupérer la liste des éléments de ma *todolist* avec un select.

4.1. Préambule : création de la table

Le script SQL suivant permet de créer la table **todolist**.

```
CREATE TABLE `todolist` (`id` INT NOT NULL
AUTO_INCREMENT , `libelle` VARCHAR( 500 ) NOT NULL ,
PRIMARY KEY ( `id` )) ENGINE = MYISAM
```

Tant qu'on est dans le SQL, on va faire un petit *insert into* pour remplir la table avec une ligne, pour que notre select renvoi quelque chose :

```
INSERT INTO `todolist` (`id`, `libelle`) VALUES (NULL ,
'Mettre le projet de demo en telechargement');
```

Ce qui nous donne :

id	libelle
1	Mettre le projet de demo en telechargement

4.2. Le script PHP de lecture

Nous allons donc créer un script PHP qui va nous retourner l'ensemble des éléments de ma *todolist* sous format XML.
L'idée est de retourner cette liste sous cette forme :

```
<datas>
        <data>
                <id>1</id>
                <libelle>Mettre le projet de démo en
téléchargement</libelle>
        </data>
        ...
</datas>
```

Voici le script PHP qui fait ca :

gettodolist.php

```
<?php
        header('Content-type: text/xml');

        $host = "localhost"; // le nom ou l'adresse du
host
        $user = "admin"; // user de la base
        $pass = ""; // mot de passe
        $connexion = mysql_connect($host,$user,$pass)
or die('Erreur de connexion');
        $bdd = "mysqldb"; // nom de la base de données
        if (!mysql_select_db($bdd,$connexion))
                return 0;
        if (!$connexion)
                return 0;
        $query = "SELECT * FROM `todolist`";

        $result = mysql_query($query);
        if (!$result)
                return 0;
        echo "<datas>";
        while ($line = mysql_fetch_assoc($result))
        {
                echo "<data>";
                $id = $line["id"];
                echo "<id>".$id."</id>";
                echo "<libelle>".
$line["libelle"]."</libelle>";
                echo "</data>";
        }
        mysql_free_result($result);     // Libération
des résultats
        echo "</datas>";

        mysql_close($connexion);        // Fermeture de
la connexion, cela ne libère pas les résultats
?>
```

Ce script commence par créer la connexion à la base de données et fait un select pour retourner tous les éléments de la liste.
Ensuite, il s'occupe de créer les balises XML en bouclant sur tous les éléments retournés par le select.
Notez l'utilisation de

gettodolist.php

```
header('Content-type: text/xml');
```

pour indiquer que l'on retourne du XML.

Cliquez ici pour tester le script (Lien58).

4.3. Téléchargement asynchrone

Nous allons donc devoir appeler ce script depuis Silverlight pour pouvoir ensuite interpréter le xml généré.

Pour ce faire, on va utiliser l'objet **WebClient** pour faire un appel asynchrone au script PHP. On utilisera la méthode **DownloadStringAsync** pour déclencher le téléchargement asynchrone et la surcharge de l'événement **DownloadStringCompletedEventHandler** permettra d'agir lors de la fin de téléchargement de la réponse renvoyée par le script PHP.

J'ai créé à cet effet une petite classe *helper* toute simple :

```
public class WebClientHelper
{
    public event DownloadStringCompletedEventHandler
DownloadComplete;
    private void OnDownloadComplete(object sender,
DownloadStringCompletedEventArgs e)
    {
        if (DownloadComplete != null)
        {
            DownloadComplete(sender, e);
        }
    }

    private readonly string _url;

    public WebClientHelper(string url)
    {
        var random = new Random();
        _url = url;
        if (_url.Contains("?"))
            _url = _url + "&trick=" + random.Next();
```

```
        else
            _url = _url + "?trick=" + random.Next();
    }

    public void Execute()
    {
        var webClient = new WebClient();
        webClient.DownloadStringCompleted +=
webClient_DownloadStringCompleted;
        webClient.DownloadStringAsync(new Uri(_url));
    }

    void webClient_DownloadStringCompleted(object
sender, DownloadStringCompletedEventArgs e)
    {
        if (e.Error == null)
        {
            OnDownloadComplete(sender, e);
        }
    }
}
```

Dans cette classe Helper, vous pouvez remarquer que je passe un paramètre en **GET** à mon script PHP qui est un nombre aléatoire *(paramètre trick)*. L'utilisation de ce paramètre permet de tromper le cache de Silverlight.
Aujourd'hui, il n'y a aucun mécanisme qui permet de définir les options de cache (cela sera surement ajouté dans la version définitive). Mon objectif est de me passer des fonctionnalités de cache, c'est pour cela que je rajoute cet argument, ce qui va forcer le script PHP a être réinterprété.

Pour l'utiliser, on fera :

```
private void ChargementDonnees()
{
        try
        {
                var helper = new
WebClientHelper(string.Format("{0}/gettodolist.php",
Config.BASEPATH));
                helper.DownloadComplete +=
helper_DownloadComplete;
                helper.Execute();
        }
        catch (Exception ex)
        {
                HtmlPage.Window.Alert(ex.Message);
        }
}
```

Vous pouvez constater que j'ai choisi de mettre l'url du serveur dans une classe de configuration :

```
public static class Config
{
        public const string BASEPATH = "http://nico-
pyright.developpez.com/tutoriel/vs2008/csharp/silverlig
htandmysql";
}
```

On associe l'événement **DownloadComplete** du helper à une méthode : helper_DownloadComplete. C'est dans cette méthode que l'on va pouvoir faire le traitement du résultat du script, c'est à dire le traitement du XML.

Remarque : n'oubliez pas de référencer l'assembly *System.Net*.

4.4. Linq to Xml

L'appel au script PHP nous renvoi donc du XML. Nous allons le parser grâce aux méthodes de **Linq To Xml**.

Dans un premier temps, nous allons créer un objet qui va représenter un élément de la *todolist* :

```
public class TodoElement
{
        public int Id { get; set; }
        public string Libelle { get; set; }
}
```

Comme on l'a vu au dessus, c'est dans la méthode helper_DownloadComplete que nous pourrons effectuer notre select avec Linq :

```
void helper_DownloadComplete(object sender,
DownloadStringCompletedEventArgs e)
{
        try
        {
                XDocument xmlElements =
XDocument.Parse(e.Result);
                var elements = from data in
xmlElements.Descendants("data")
                                        select new
TodoElement
                                        {
                                                Id =
(int)data.Element("id"),
                                                Libe
lle = ((string)data.Element("libelle")).Trim()
                                        };
                ListeTodo.ItemsSource = elements;
        }
        catch (Exception ex)
        {
                HtmlPage.Window.Alert(ex.Message);
        }
}
```

Remarque : N'oubliez pas de rajouter la référence à *System.Xml.Linq*.

Le résultat de l'exécution du script PHP, s'il est correct, est disponible dans **e.Result**. On peut donc s'en servir pour le parser avec l'object **XDocument**. Nous pourrons alors faire notre requête Linq.
Notez qu'on affecte la liste des éléments construits à la propriété **ItemSource** d'un objet ListeTodo.
Il s'agit d'une ListBox qui sera déclarée dans le XAML de notre page, nous y reviendrons plus tard.

Vous avez donc vu, à travers ces différentes étapes, comment on pouvait faire une requête SQL de type *select* dans notre base MySQL.

5. Mise à jour de la table

De la même façon, on veut pouvoir effectuer d'autres opérations sur la table, comme un **insert into** ou un **update**, afin de rajouter ou de supprimer des éléments dans notre *todolist*. Pour ça, on va passer encore une fois par un script PHP. Sauf que cette fois ci, nous allons devoir lui passer des paramètres. Pour ce faire, nous allons les lui envoyer **en POST**.

5.1. Script PHP d'ajout d'un élément dans la liste

add.php

```
<?php
        $host = "localhost"; // le nom ou l'adresse du
host
        $user = "admin"; // user de la base
        $pass = ""; // mot de passe
        $connexion = mysql_connect($host,$user,$pass)
```

```
or die('Erreur de connexion');
        $bdd = "mysqldb"; // nom de la base de données
        if (!mysql_select_db($bdd,$connexion))
                return 0;
        if (!$connexion)
        {
                echo "ERR1";
                return 0;
        }

        if (isset($_POST["data"]))
        {
                $libelle =
mysql_real_escape_string($_POST["data"]);
        $query = "INSERT INTO `todolist` (`id`,
`libelle`) VALUES (NULL , '".$libelle."')";

                $result = mysql_query($query);
                if (!$result)
                        echo "ERR2";
        }
        else
                echo "ERR";

        mysql_close($connexion);        // Fermeture de
la connexion, cela ne libère pas les résultats
?>
```

Dans un premier temps, ce script se connecte à la base de données. Ensuite, il vérifie s'il y a dans les données postées quelque chose qui l'intéresse. Si c'est le cas, on l'insère dans la table todolist.
Ce qui a pour but d'ajouter un nouvel élément dans la *todolist*.

5.2. Envoi de données en POST

Le principe est d'utiliser l'objet **HttpWebRequest** *(n'oubliez pas de rajouter la référence à System.Net)* et d'envoyer les données de manière asynchrone en POST via **BeginGetRequestStream**.
J'utilise ici un helper inspiré du site d'Albert Cameron (Lien59).

Voici son implémentation :

```
public class HttpHelper
{
    private HttpWebRequest Request { get; set; }
    public Dictionary<string, string> PostValues { get;
private set; }

    public event HttpResponseCompleteEventHandler
ResponseComplete;
    private void
OnResponseComplete(HttpResponseCompleteEventArgs e)
    {
        if (ResponseComplete != null)
        {
            ResponseComplete(e);
        }
    }

    public HttpHelper(Uri requestUri, string method,
params KeyValuePair<string, string>[] postValues)
    {
        Request =
(HttpWebRequest)WebRequest.Create(requestUri);
        Request.ContentType = "application/x-www-form-
urlencoded";
        Request.Method = method;
        PostValues = new Dictionary<string, string>();
        if (postValues != null && postValues.Length >
0)
        {
            foreach (var item in postValues)
            {
                PostValues.Add(item.Key, item.Value);
            }
        }
    }

    public void Execute()
    {
        Request.BeginGetRequestStream(BeginRequest,
this);
    }

    private static void BeginRequest(IAsyncResult ar)
    {
        var helper = ar.AsyncState as HttpHelper;
        if (helper != null)
        {
            if (helper.PostValues.Count > 0)
            {
                using (var writer = new
StreamWriter(helper.Request.EndGetRequestStream(ar)))
                {
                    foreach (var item in
helper.PostValues)
                    {
                        writer.Write("{0}={1}&",
item.Key, item.Value);
                    }
                }
            }
            helper.Request.BeginGetResponse(BeginRespon
se, helper);
        }
    }

    private static void BeginResponse(IAsyncResult ar)
    {
        var helper = ar.AsyncState as HttpHelper;
        if (helper != null)
        {
            var response =
(HttpWebResponse)helper.Request.EndGetResponse(ar);
            if (response != null)
            {
                var stream =
response.GetResponseStream();
                if (stream != null)
                {
                    using (var reader = new
StreamReader(stream))
                    {
                        helper.OnResponseComplete(new
HttpResponseCompleteEventArgs(reader.ReadToEnd()));
                    }
                }
            }
        }
    }
}

public delegate void
HttpResponseCompleteEventHandler(HttpResponseCompleteEv
entArgs e);
public class HttpResponseCompleteEventArgs : EventArgs
{
    public string Response { get; set; }

    public HttpResponseCompleteEventArgs(string
response)
    {
        Response = response;
    }
```

```
}
```

Pour l'utiliser :

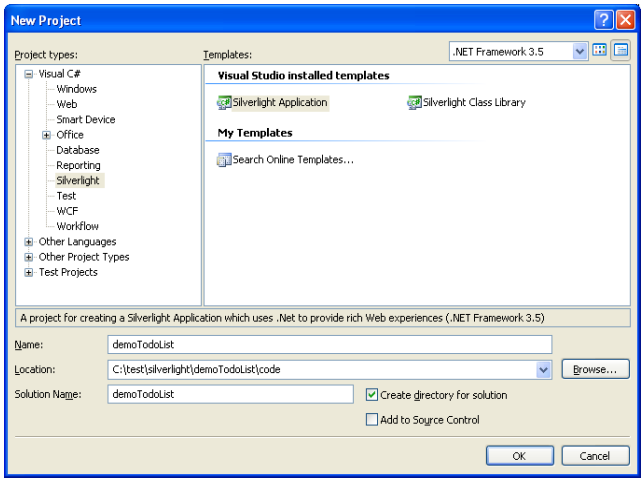
```
var helper = new HttpHelper(new Uri(string.Format("{0}/
add.php", Config.BASEPATH)), "POST",
                                new
KeyValuePair<string, string>("data",
nouvelleTache.Text));
helper.ResponseComplete += AddComplete;
helper.Execute();
```

```
private void AddComplete(HttpResponseCompleteEventArgs
e)
{
        string retour = e.Response;
        if (retour == "ERR")
                HtmlPage.Window.Alert("Erreur dans
l'ajout");
        else
        {
                if (retour == "ERR1")
                        HtmlPage.Window.Alert("Problème
de connexion à la base de données");
                else
                {
                        if (retour == "ERR2")
                                HtmlPage.Window.Alert("
Problème d'insertion");
                        else
                                ChargementDonnees();
                }
        }
}
```

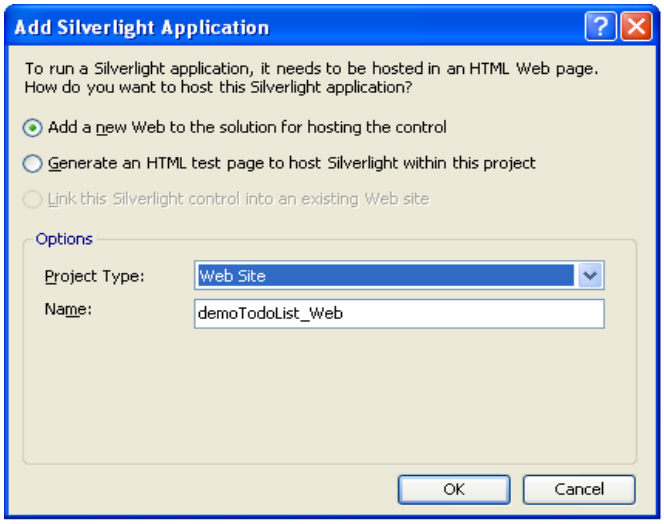
La méthode AddComplete est appelée lorsque l'envoi des données en POST est terminé. On récupère le retour de cet appel dans **e.Response**, ce qui me permettra de vérifier que tout s'est bien passé.
Puis on rappelle le chargement des données pour prendre en compte la nouvelle donnée.

6. L'application : une todo liste

Je commence par créer un nouveau projet de type **Silverlight Application**.



Ensuite je choisis d'utiliser un nouveau site web qui contiendra le contrôle silverlight.



NB : Si vous n'utilisez pas ce type de projet, vous aurez une exception de type **"security error"** lors de l'utilisation de l'objet WebClient.

Retrouvez la suite de l'article de Nico-pyright(c) en ligne : Lien60

Les derniers tutoriels et articles

La programmation des sockets bruts sous Windows

Ce tutoriel va vous apprendre la programmation des sockets bruts (SOCK_RAW) sous Windows en langage C au sein d'un environnement TCP/IP.

1. Introduction

La dernière fois nous avons vu comment programmer les sockets en mode connecté (SOCK_STREAM) et en mode non connecté (SOCK_DGRAM). Cette fois-ci nous allons nous intéresser aux sockets bruts (**SOCK_RAW**) qui permettent de travailler à plus bas niveau avec les protocoles d'une famille donnée, dans notre cas : la suite de protocoles TCP/IP (**PF_INET**), ou même d'inventer de nouveaux protocoles mais là n'est pas notre objectif. Nous travaillerons principalement sous Windows mais il ne sera pas du tout difficile d'adapter nos programmes pour qu'ils puissent compiler pour d'autres plateformes. Entrons maintenant dans le vif du sujet.

2. Compléments sur les structures et les champs de bits

2.1. Introduction

La manipulation des en-têtes TCP, IP, etc. nécessitent une bonne compréhension des propriétés des structures et des champs de bits. La difficulté vient du fait que beaucoup de ces propriétés ne sont pas définies par la norme. Afin de mieux nous concentrer sur notre véritable problème plutôt que sur des détails d'implémentation du C, nous utiliserons, dans ce tutoriel, le compilateur Visual C++ (version 6 ou supérieure) étant donné que nous avons déjà fait le choix de travailler sous Windows. Tous les exemples de cette page ont été compilés puis testés avec Visual C++ 2005, Borland C++ Compiler 5.5 et gcc (GCC) 3.4.5 sous Windows XP Professionnel Service Pack 2.

2.2. Alignement

La norme du langage C stipule que les membres d'une structure doivent apparaitre en mémoire dans le même ordre dans lequel ils ont été déclarés. Cependant, c'est une erreur de croire que les membres d'une structure soient toujours collés (serrés) entre-eux autrement dit que la taille d'une structure soit toujours égale à la somme des tailles de chacun de ses membres. Cela vient du fait que les objets ne sont pas stockés en mémoire de façon aléatoire. En effet, pour des raisons de performance (cela est même requis par certains processeurs), chaque objet doit être alloué en mémoire à une adresse p telle que p soit un multiple entier d'un entier n appelé **align-requirement** (alignment requirement) de l'objet en question. Cette contrainte est connue dans la littérature française sous le nom de **contrainte d'alignement**.

Pour les types de base (char, short, int, double, etc.), ce nombre est égal à la taille de ce type. Par exemple, la taille d'un int étant de 32 bits (4 octets) sur un processeur Intel 32 bits, n est donc de 4, ce qui signifie qu'une variable de type int doit commencer à une adresse qui est multiple de 4. Pour une structure (ou un tableau ou une union), la contrainte d'alignement est par défaut celle du membre dont l'alignement requis est le plus grand. De plus, chaque membre de la structure commence à une adresse qui est soit multiple de sa taille, soit multiple du nombre imposé par la contrainte d'alignement de la structure, en prenant le minimum. Cela permet de garantir que chaque membre de la structure soit correctement aligné pour le processeur. Pour illustrer ce principe, considérons la structure suivante :

```
struct s {
  char  c; /* 1 octet  */
  int   n; /* 4 octets */
};

struct s x;
```

Avec les réglages par défaut du compilateur, on a sizeof(x) == 8 et non 5. En effet, la contrainte d'alignement de la structure est de 4 car sizeof(n) == 4, n étant le membre dont l'alignement requis ici est le plus grand (4 contre 1 pour c) donc : x doit commencer à une adresse multiple de 4 (mais en fait ce détail ne nous intéresse pas ici), c doit commencer à une adresse multiple de 1 ou de 4 donc 1 (autrement dit peut commencer à n'importe quelle adresse) car 1 est le minimum des deux et n doit commencer à une adresse multiple de 4. La taille de la structure est donc de 8.

Considérons maintenant la structure suivante :

```
struct s {
  char  c; /* 1 octet  */
  short u; /* 2 octets */
  int   n; /* 4 octets */
};

struct s x;
```

La taille de x ici est toujours de 8 octets car c doit commencer à une adresse multiple de 1, u à une adresse multiple de 2 et n à une adresse multiple de 4, avec comme tailles respectives 1, 2 et 4. Si on considère cependant l'exemple suivant :

```
struct s {
  char  c; /* 1 octet  */
  char  d; /* 1 octet  */
  short u; /* 2 octets */
  int   n; /* 4 octets */
};

struct t {
  char  c; /* 1 octet  */
  short u; /* 2 octets */
  char  e; /* 1 octet  */
  int   n; /* 4 octets */
};

struct s x;
struct t y;
```

La taille de x sera de 8 tandis que celle de y sera de 12 ! Vous en savez je crois assez pour expliquer ce phénomène.

2.3. Les champs de bits

Les champs de bits sont des bits consécutifs d'un objet de type entier. L'objet en question doit cependant avoir été encapsulé dans une structure. Par exemple :

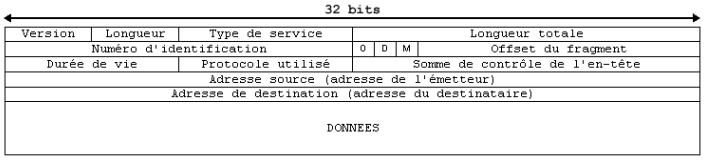
```
struct t {
  unsigned a;
  unsigned m : 23;
  unsigned e : 8;
  unsigned s : 1;
};

struct t x;
```

En faisant abstraction de la structure, y est une structure composée de deux entiers non signés - a et un autre décomposé en 3 champs de bits à savoir : m (bits 0 à 22), e (bits 23 à 30) et s (bit 31). La taille de x est de 8 octets.

La norme du langage C stipule qu'un champ de bits doit être créé à l'intérieur d'un int ou unsigned int mais autorise une implémentation particulière à supporter d'autres types. Ces derniers doivent cependant être de type entier. La norme laisse également au soin de chaque implémentation de définir l'ordre dans lequel ces champs sont rangés à l'intérieur de leur support. Dans l'implémentation du C de Microsoft, un champ de bits peut être créé au sein de n'importe quel type entier, signé ou non et ils sont rangés allant du bit du poids le plus faible au bit de poids le plus fort de leur support.

3. IP : Le protocole de l'Internet

Comme nous les savons déjà le données qui circulent sur Internet se présentent sous la forme de paquets IP, constitué d'un en-tête (contenant entre autres l'adresse du destinataire et celle de l'émetteur) immédiatement suivi des données à transporter. Ces données proviennent de la couche juste au dessus de IP c'est-à-dire du protocole **TCP** ou **UDP** ou même d'un protocole de même niveau comme **ICMP** par exemple. Ces données se présentent également sous forme de paquet (on parle donc de paquet TCP, UDP ou ICMP ...) constitué d'un en-tête suivi également des données originales. Les sockets bruts permettent entre autres de composer soi-même un paquet IP et de le soumettre au réseau c'est pourquoi il est important de connaître le format d'un tel paquet. Vous avez donc deviné, cela permet par exemple de composer un paquet avec une adresse source (adresse de l'émetteur) falsifiée ! Le format d'un paquet IP, dans la version 4 (IPv4), est le suivant :



Avant d'expliquer le rôle de chaque champ, passons tout d'abord par la définition la structure C représentant l'en-tête d'un paquet IP :

Fichier : ip.h

```
#ifndef H_IP_H

#define H_IP_H

typedef struct ipheader {
  UCHAR  hlen : 4, version : 4;
  UCHAR  tos;
  USHORT tot_len;
  USHORT id;
  USHORT offset : 13, flags : 3;
  UCHAR  ttl;
  UCHAR  protocole;
  USHORT somme;
  ULONG  ip_source;
  ULONG  ip_dest;
} IPHEADER;

#endif
```

Voici donc à présent la signification de chacun de ces champs :

- **Version** : indique la version du protocole utilisé. Il faut mettre 4 car on utilise la version 4 de ce protocole.
- **Longueur** : contient le nombre de mot de 32 bits constituant l'en-tête. Ce champ contient donc généralement 5 mais en fait, l'en-tête peut également comporter des "options" et dans ce cas, sa taille devient supérieure à 5 (en fonction de la taille des options).
- **Type de service** : ce champ permet d'indiquer le type de service exigé par le client (l'utilisateur). Le service doit être supporté par le routeur. On peut tout simplement mettre zéro si aucun service particulier n'est exigé.
- **Longueur totale** : la longueur (mesurée en octets) du paquet IP. Un paquet IP peut donc faire tout au plus 65536 octets (64 Ko). En fait, chaque type de support de transmission est caractérisé par la taille du plus gros paquet que celui-ci peut transporter. Si jamais la taille d'un paquet dépasse le maximum autorisé, IP le découpe en de plus petits paquets pouvant chacun être accepté par le support. Ce processus est appelé la **fragmentation** des paquets. A l'arrivée, les fragments seront raccollés pour reconstituer le paquet initial.
- **Numéro d'identification** : il s'agit d'un numéro attribué à chaque frament afin de permettre leur réassemblage. Autrement dit les fragments d'un paquet donné doivent porter le même numéro.
- **Flags (0, D et M)** : ce champ est constitué de 3 bits à savoir **0** : inutilisé, **D** (**D**o not fragment) : si mis à 1, indique que le paquet ne doit pas être fragmenté (si jamais la taille du paquet dépasse le maximum autorisé, le paquet ne sera pas envoyé) et **M** (**M**ore fragments) : si mis à 1, indique qu'il y a encore d'autres fragments (ce qui signifie que le paquet lui-même est également un fragment). Si ce bit vaut 0, cela signifie que le paquet est le dernier des fragments du paquet à reconstituer ou tout simplement que le paquet n'a pas subi de fragmentation.
- **Offset** : indique la position (offset) à partir de laquelle les données actuelles se trouvait dans le paquet initial. Cette position n'est pas cependant exprimée en octets car la taille d'un paquet IP est codée sur 16 bits alors que ce champ n'utilise que 13 bits. Les données doivent donc commencer à une position multiple de 8 octets (0 pour le premier fragment), nombre alors choisi comme unité de ce champ.
- **Durée de vie (TTL)** : indique d'une certaine manière le nombre maximal de routeurs que le paquet peut traverser, bref : sa "durée de vie". Ce champ est décrémenté chaque fois que le paquet tombe sur un routeur. Si après décrémentation la valeur de ce champ est 0, le routeur détruira le paquet. Comme nous pouvons le constater, cela permet de tuer un paquet qui met trop de temps à atteindre la destination.
- **Protocole** : permet de savoir quel type de paquet (TCP ? UDP ? ICMP ? ...) est encapsulé dans le paquet IP. Ce champ vaut **6** (**IPPROTO_TCP**) si le paquet encapsulé provient de TCP, **17** (**IPPROTO_UDP**) s'il provient

d'UDP et **1** (**IPPROTO_ICMP**) s'il provient d'ICMP.

- **Somme de contrôle (Checksum)** : valeur permettant de vérifier l'integrité de l'en-tête. Elle doit être égale au complément à 1 d'une somme spéciale de tous les mots de 16 bits de l'en-tête, calculée en mettant ce champ à 0. Pour calculer cette somme : on fait la somme de tous les mots de 16 bits de l'en-tête, on sauve le résultat (32 bits). Si la taille de l'en-tête n'est pas un multiple de 16 bits, prendre l'octet restant puis l'ajouter au résultat précédent. Tant que le résultat ne tient pas sur 16 bits, calculer la somme de tous les mots de 16 bits de cette somme. Voir plus bas pour un exemple de fonction calculant le checksum d'un message contenu dans un buffer dont l'adresse de base et la taille sont passées en arguments.
- **Adresse source** : adresse IP de la source.
- **Adresse de destination** : adresse IP de la destination.

La fonction suivante permet le calcul du checksum tel que défini dans la spécificiation du protocole IP :

Fonction : checksum

```
USHORT checksum(void * lpData, size_t size)
{
  USHORT * t = lpData;
  DWORD somme = 0;
  size_t i, n = size / sizeof(USHORT);

  for(i = 0; i < n; i++)
    somme += t[i];

  if (size % 2)
  {
    UCHAR * u = lpData;
    somme += u[size - 1];
  }

  while (HIWORD(somme))
    somme = HIWORD(somme) + LOWORD(somme);

  return LOWORD(~somme);
}
```

4. Les sockets bruts

4.1. Généralités

Les sockets bruts sont tout simplement des sockets de type **SOCK_RAW**. Dans la suite, nous verrons comment utiliser ces sockets avec TCP/IP pour pouvoir travailler à bas niveau avec les protocoles de cette suite. Avec ce type de socket, on a plus de choix concernant la valeur qu'on peut spécifier dans le paramètre *protocol* de la fonction socket. Pour un socket TCP/IP, ceci peut être par exemple **IPPROTO_TCP** pour un socket TCP, **IPPROTO_UDP** pour un socket UDP, **IPPROTO_ICMP** pour un socket ICMP, **IPPROTO_IP** pour un socket supportant tous les protocoles de la suite ou encore **IPPROTO_RAW** pour pouvoir utiliser un protocole personnalisé. Voici par exemple comment créer un socket capable d'envoyer et de recevoir des paquets IP transportant un message UDP et de tels paquets uniquement :

```
SOCKET s;

s = socket(PF_INET, SOCK_RAW, IPPROTO_UDP);
if (s == INVALID_SOCKET)
  /* La fonction socket a echoue */ ;
else
{
  /* On a reussi */

  closesocket(s);
}
```

Vous devez normalement être administrateur pour pouvoir créer un socket de type SOCK_RAW. En outre, il est important de savoir que pour pouvoir composer un paquet IP soi-même, il faut activer l'option **IP_HDRINCL** (déclaré dans **ws2tcpip.h**) du niveau **IPPROTO_IP**, avec **setsockopt** bien sûr.

```
DWORD sockopt = TRUE;

if (setsockopt(s, IPPROTO_IP, IP_HDRINCL, (char
*)&sockopt, sizeof(sockopt)) == SOCKET_ERROR)
  /* La fonction setsockopt a echoue */ ;
else
{
  /* On continue */
}
```

Si l'option **IP_HDRINCL** n'est pas activée alors nous ne pouvons nous contenter que de pouvoir composer le paquet à soumettre à IP, sans le pouvoir de composer le paquet IP lui-même. A noter cependant qu'en reception (recv ...), c'est toujours un paquet IP qui est copié dans le buffer que cette option ait été activée ou non. Il faut donc s'assurer que le buffer soit assez grand pour pouvoir contenir n'importe quel paquet IP si on ne veut pas se faire des ennuis. En outre, sachez que l'utilisation des protocoles de niveau message, c'est-à-dire TCP ou UDP, nécessite toujours l'activation de l'option IP_HDRINCL. Seuls les protocoles de même niveau que IP, comme ICMP ou IGMP par exemple, peuvent être utilisés avec ou sans cette option.

Pour ce qui est des opérations de lecture/écriture, ce sont les fonctions **sendto** et **recvfrom** qu'il faut utiliser car send et recv ne sont adaptées que pour les sockets connectés. Dans sendto, il faut que l'adresse spécifiée dans la structure pointée par *to* corresponde à l'adresse de destination spécifiée dans dans l'en-tête du paquet IP sinon la fonction échouera. Dans recvfrom, l'adresse source spécifiée dans l'en-tête IP sera copiée dans la structure pointée par *from*.

Et enfin, sachez que depuis Windows XP SP 2 (donc valable également pour Vista), Windows ne permet pas la composition manuelle de paquets TCP. La raison de cette limitation est que la plupart des attaques utilisées sur Internet sont basées sur l'exploitation de ce protocole ... D'autre part, l'adresse source d'un paquet encapsulant UDP doit être une adresse existante sur le réseau, cela afin de limiter la possibilité de falsification d'adresse. Si vous voulez donc programmer les sockets bruts sans aucune limitation, vous devriez utiliser Windows XP sans service pack par exemple ou encore changer carrément de système (Linux fait bien l'affaire). Dans ce tutoriel, nous faisons l'hypothèse que nous somme sous Windows XP SP 2 et qu'il faut donc tenir compte de ces limitaions.

4.2. Le protocole UDP

Nous avons déjà eu l'occasion de parler de ce protocole plusieurs fois, en particulier dans notre introduction aux sockets. UDP est un des protocoles les plus simples, son en-tête ne comporte que 4 champs (contre 16 pour le protocole TCP !) immédiatement suivi des données : le numéro de **port source** (16 bits), le numéro de **port de destination** (16 bits), la **taille** (mesurée en octets) du paquet et la **somme de contrôle** d'un "pseudo en-tête" UDP. Dans IPv4, cette somme est cependant facultative, UDP étant un protocole non fiable, et on peut tout simplement le laisser à 0 (si une valeur différente de 0 est spécifiée, le réseau considèrera que ce champ doit être pris en compte). Pour plus de détails concernant cette structure, référez vous à la spécification de ce protocole.
Passons maintenant à la définition de la structure C

correspondante :

Fichier : udp.h

```
#ifndef H_UDP_H

#define H_UDP_H

typedef struct udpheader {
  USHORT port_source;
  USHORT port_dest;
  USHORT tot_len;
  USHORT somme;
} UDPHEADER;

#endif
```

L'exemple suivant montre comment composer un paquet IP transoprtant un paquet UDP en utilisant les sockets bruts. Le message sera "Bonjour." et sera envoyé sur le port 5050 de la machine locale. Il est donc supposé qu'on a de l'autre côté un serveur UDP en attente du message.

Fichier : udp_send.c

```
#include <stdio.h>
#include <winsock2.h>
#include <ws2tcpip.h>
#include "ip.h"
#include "udp.h"
#include <stdlib.h>

#define MAX_MESSAGE 100
#define IP_SOURCE "127.0.0.1"
#define IP_DEST "127.0.0.1"
#define PORT_SOURCE 5050
#define PORT_DEST 5050

USHORT checksum(void * lpData, size_t size);

int main()
{
  WSADATA wsaData;

  if (WSAStartup(MAKEWORD(2, 0), &wsaData) != 0)
    fprintf(stderr, "La fonction WSAStartup a
echoue.\n");
  else
  {
    SOCKET s;

    s = socket(PF_INET, SOCK_RAW, IPPROTO_UDP);
    if (s == INVALID_SOCKET)
      fprintf(stderr, "La fonction socket a
echoue.\n");
    else
    {
      DWORD sockopt = TRUE;

      if (setsockopt(s, IPPROTO_IP, IP_HDRINCL, (char
*)&sockopt, sizeof(sockopt)) == SOCKET_ERROR)
        fprintf(stderr, "La fonction setsockopt a
echoue.\n");
      else
      {
        char message[MAX_MESSAGE] = "Bonjour.", * buf;
        USHORT message_len, udp_len, ip_len;

        message_len  = (USHORT)(strlen(message) + 1); /
* Longueur du message */
        udp_len      = (USHORT)(sizeof(UDPHEADER) +
message_len); /* Longueur du paquet UDP */
        ip_len       = (USHORT)(sizeof(IPHEADER) +
udp_len); /* Longueur du paquet IP */
        buf = malloc(ip_len);
        if (buf == NULL)
          fprintf(stderr, "La fonction malloc a
echoue.\n");
        else
        {
          SOCKADDR_IN dest;
          IPHEADER ip;
          UDPHEADER udp;

          /* Remplissage de la structure ip (en-tete du
paquet) */

          /* Rappel : Attention a l'ordre des octets
(pour les mots de plus de 1 octet).         */
          /*          Utilisez htons et htonl pour
mettre les shorts et les longs au bon format. */
          /*          Cela ne concerne pas les chars
car la taille d'un char est de 1 octet.     */

          ip.version    = 4;
          ip.hlen       = 5; /* sizeof(ip) / 4 */
          ip.tos        = 0;
          ip.tot_len    = htons(ip_len);
          ip.id         = htons(1);
          ip.flags      = 0;
          ip.offset     = 0;
          ip.ttl        = 100;
          ip.protocole  = IPPROTO_UDP;
          ip.somme      = 0;
          ip.ip_source  = inet_addr(IP_SOURCE);
          ip.ip_dest    = inet_addr(IP_DEST);

          ip.somme      = checksum(&ip, sizeof(ip));

          udp.port_source  = htons(PORT_SOURCE);
          udp.port_dest    = htons(PORT_DEST);
          udp.tot_len      = htons(udp_len);
          udp.somme        = 0;

          memcpy(buf, &ip, sizeof(ip));
          memcpy(buf + sizeof(ip), &udp, sizeof(udp));
          memcpy(buf + sizeof(ip) + sizeof(udp),
message, message_len);

          ZeroMemory(&dest, sizeof(dest));
          dest.sin_family       = AF_INET;
          dest.sin_addr.s_addr  = inet_addr(IP_DEST);
          dest.sin_port         = htons(PORT_DEST);

          if (sendto(s, buf, ip_len, 0, (SOCKADDR
*)&dest, sizeof(dest)) == SOCKET_ERROR)
            fprintf(stderr, "La fonction sendto a
echoue.\n");
          else
            printf("Message envoye !\n");

          free(buf);
        }
      }

      closesocket(s);
    }

    WSACleanup();
  }

  return 0;
}

USHORT checksum(void * lpData, size_t size)
{
```

```
  USHORT * t = lpData;
  DWORD somme = 0;
  size_t i, n = size / sizeof(USHORT);

  for(i = 0; i < n; i++)
    somme += t[i];

  if (size % 2)
  {
    UCHAR * u = lpData;
    somme += u[size - 1];
  }

  while (HIWORD(somme))
    somme = HIWORD(somme) + LOWORD(somme);

  return LOWORD(~somme);
}
```

4.3. Le protocole ICMP

ICMP (**I**nternet **C**ontrol **M**essage **P**rotocol) est un protocole faisant partie de la suite TCP/IP, permettant d'obtenir des informations sur la structure du réseau et de gérer les erreurs de transmission. Par exemple, lorsqu'un paquet IP a expiré (le TTL a atteint la valeur critique 0), le routeur qui a causé cette mort du paquet envoie à l'émetteur un message (un paquet ICMP) indiquant que le paquet a été détruit car sa durée de vie a atteint la limite. Les codes d'erreurs de ce protocole sont hélas trop nombreux pour pouvoir être détaillés ici. Avant de continuer avec les possibilités offertes par ICMP, voyons tout d'abord à quoi ressemble un message ICMP.

Un message ICMP est composé d'un champ indiquant le **type** du message (8 bits), d'un champ indiquant le **code** (qui peut être par exemple un code d'erreur dans le cas d'un message d'erreur) du message (8 bits), de la **somme de contrôle** du message (16 bits) et des **données** (de longueur variable et dont la signification dépend du type du message). On peut donc dire en quelque sorte que tout est inclus dans l'en-tête et que la structure d'un en-tête ICMP varie selon le type du message.

Prenons l'exemple d'une requête **ping**. Une requête ping est un message ICMP qui sollicite le destinataire de ce message à renvoyer ce message. Il est important de comprendre que les réponses ICMP, de même que les messages d'erreur, sont générées "automatiquement", c'est-à-dire sans l'intervention d'une application (de niveau application) prévue à cet effet. En d'autres termes, ces types de messages sont générés par le pilote de l'interface de la machine (l'ordinateur ou le routeur) elle-même. En effet, ICMP est un protocole de niveau réseau comme IP et non un protocole de la couche transport ou application.

Revenons au ping. Un ping est un message ICMP de type **8** (**Echo request**) transportant des données permettant par exemple de le reconnaître lorsqu'il sera renvoyé par l'hôte distant. Si l'hôte a reçu le message, donc les données, il va les retourner à l'envoyeur à l'intérieur d'un message de type **0** (**Echo reply**). Voici donc la structure de l'en-tête commun à tous les messages ICMP :

Fichier : icmp.h

```
#ifndef H_ICMP_H

#define H_ICMP_H

typedef struct icmpheader {
  UCHAR  type;
  UCHAR  code;
  USHORT somme;
} ICMPHEADER;

#endif
```

Et voici le format (personnalisé) des messages échangés lors d'un ping :

Fichier : ping.h

```
#ifndef H_PING_H

#define H_PING_H

typedef struct icmppingmessage {
  /* 'En-tete' ICMP */
  UCHAR  type;
  UCHAR  code;
  USHORT somme;

  /* 'Donnees' (definis par le programmeur) */
  DWORD  id; /* Ce champ nous permettra de reconnaitre
notre message */
  DWORD  timestamp; /* Celui-ci nous permettra de
savoir quand ce message a ete envoye */
} ICMPPINGMESSAGE;

#define ICMP_ECHO_REQ 8
#define ICMP_ECHO_REPLY 0

#endif
```

Nous aurons également besoin de fonctions permettant de connaître l'adresse d'un hôte dont le nom est donné afin que notre programme soit plus simple d'emploi. N'oubliez pas non plus qu'on ne peut pas mettre 127.0.0.1 dans le champ adresse source de l'en-tête IP sauf dans le cadre d'un test local sinon le message ne nous reviendra pas (il va être "avalé" par le destinataire !). Voici les deux fonctions qui nous seront utiles :

Fonctions : resolve_addr, resolve_computer_addr

```
/* Cette fonction permet de determiner l'adresse IP
d'un hote dont le nom est fourni en argument */

HOSTENT * resolve_addr(char * name, ULONG * p_addr)
{
  HOSTENT * h = gethostbyname(name);

  if (h != NULL)
    memcpy(p_addr, h->h_addr_list[0], sizeof(ULONG));

  return h;
}

/* Cette fonction permet de determiner l'adresse IP de
l'ordinateur */

HOSTENT * resolve_computer_addr(ULONG * p_addr)
{
  HOSTENT * h = NULL;
  char computer_name[100];

  if (gethostname(computer_name, sizeof(computer_name))
== 0)
    h = resolve_addr(computer_name, p_addr);

  return h;
}
```

Enfin, le programme ne va pas attendre indéfiniment le retour du message. Si au bout d'un certain temps fixé celui-ci ne revient toujours pas, on abandonne l'attente et affiche un message indiquant que le message a mis trop de temps pour revenir.

Voici donc un petit programme qui envoie une requête ping à la destination spécifiée et reste en attente de la réponse pendant un intervalle de temps donné. Pour réduire la taille de notre fichier source, les définitions des fonctions resolve_addr, resolve_computer_addr et checksum n'ont pas été incluses dans ce fichier.

Fichier : echo_req.c

```
#include <stdio.h>
#include <winsock2.h>
#include <ws2tcpip.h>
#include "ip.h"
#include "ping.h"
#include <stdlib.h>

#define REMOTE "localhost"
#define ICMP_PING_TIMEOUT 2000

HOSTENT * resolve_addr(char * name, ULONG * p_addr);
HOSTENT * resolve_computer_addr(ULONG * p_addr);
USHORT checksum(void * lpData, size_t size);

int main()
{
  WSADATA wsaData;

  if (WSAStartup(MAKEWORD(2, 0), &wsaData) != 0)
    fprintf(stderr, "La fonction WSAStartup a
echoue.\n");
  else
  {
    SOCKET s;

    s = socket(PF_INET, SOCK_RAW, IPPROTO_ICMP);
    if (s == INVALID_SOCKET)
      fprintf(stderr, "La fonction socket a
echoue.\n");
    else
    {
      DWORD sockopt;

      sockopt = TRUE;

      if (setsockopt(s, IPPROTO_IP, IP_HDRINCL, (char
*)&sockopt, sizeof(sockopt)) == SOCKET_ERROR)
        fprintf(stderr, "La fonction setsockopt a
echoue (IP_HDRINCL).\n");
      else
      {
        sockopt = ICMP_PING_TIMEOUT;

        if (setsockopt(s, SOL_SOCKET, SO_RCVTIMEO,
(char *)&sockopt, sizeof(sockopt)) == SOCKET_ERROR)
          fprintf(stderr, "La fonction setsockopt a
echoue (SO_RCVTIMEO).\n");
        else
        {
          char * buf;
          USHORT ip_len = (USHORT)(sizeof(IPHEADER) +
sizeof(ICMPPINGMESSAGE));

          buf = malloc(ip_len);
          if (buf == NULL)
            fprintf(stderr, "La fonction malloc a
echoue.\n");
          else
          {
            ULONG ip_source;

          if (resolve_computer_addr(&ip_source) ==
NULL)
            fprintf(stderr, "La fonction
resolve_computer_addr a echoue.\n");
          else
          {
            ULONG ip_dest;
            if (resolve_addr(REMOTE, &ip_dest) ==
NULL)
              fprintf(stderr, "Impossible de trouver
l'hote '%s'.\n", REMOTE);
            else
            {
              SOCKADDR_IN remote;
              int remote_len = (int)sizeof(remote);
              IPHEADER ip;
              ICMPPINGMESSAGE icmp;

              ip.version    = 4;
              ip.hlen       = 5;
              ip.tos        = 0;
              ip.tot_len    = htons(ip_len);
              ip.id         = htons(1);
              ip.flags      = 0;
              ip.offset     = 0;
              ip.ttl        = 100;
              ip.protocole  = IPPROTO_ICMP;
              ip.somme      = 0;
              ip.ip_source  = ip_source;
              ip.ip_dest    = ip_dest;

              ip.somme      = checksum(&ip,
sizeof(ip));

              icmp.type       = ICMP_ECHO_REQ;
              icmp.code       = 0;
              icmp.somme      = 0;
              icmp.id         =
GetCurrentProcessId();
              icmp.timestamp  = GetTickCount();

              icmp.somme      = checksum(&icmp,
sizeof(icmp));

              memcpy(buf, &ip, sizeof(ip));
              memcpy(buf + sizeof(ip), &icmp,
sizeof(icmp));

              ZeroMemory(&remote, sizeof(remote));
              remote.sin_family       = AF_INET;
              remote.sin_addr.s_addr  = ip_dest;

              printf("Envoi d'une requete ping sur %s
avec %hu octets de donnees.\n",
              inet_ntoa(remote.sin_addr), ip_len);

              if (sendto(s, buf, ip_len, 0, (SOCKADDR
*)&remote, remote_len)
              == SOCKET_ERROR)
                fprintf(stderr, "La fonction sendto a
echoue.\n");
              else
              {
                printf("Attente de la reponse.\n");

                if (recvfrom(s, buf, ip_len, 0,
(SOCKADDR *)&remote, &remote_len)
                == SOCKET_ERROR)
                {
                  if (WSAGetLastError() ==
WSAETIMEDOUT)
                    printf("Delai d'attente depasse.\
n");
                  else
                    fprintf(stderr, "La fonction recv
a echoue.\n");
                }
```

```
                else
                {
                  DWORD delta_t = GetTickCount() -
icmp.timestamp;

                  ZeroMemory(&ip, sizeof(ip));
                  ZeroMemory(&icmp, sizeof(icmp));

                  memcpy(&ip, buf, sizeof(ip));
                  memcpy(&icmp, buf + (ip->hlen * 4),
sizeof(icmp));

                if (icmp.id != GetCurrentProcessId())
                   printf("Recu : paquet
etranger.\n");
                  else
                  {
                    switch(icmp.type)
                    {
                      case ICMP_ECHO_REPLY:
                        printf("Temps de reponse =
%lu ms\n", delta_t);
                        break;

                      default:
                        printf("Recu : paquet
inattendu.\n");
                        break;
                    }
                  }
                }
              }
            }
          }
        }

        free(buf);
      }
    }
  }

  closesocket(s);
 }

  WSACleanup();
}

return 0;
}
```

Bien entendu, un programme bien abouti ne se contentera pas d'envoyer une seule requête à l'hôte distant et d'afficher le temps de réponse mais maintenant que avez compris le principe, vous pouvez développer des applications plus complexes.

Retrouvez la suite de l'article de Jessee Edouard en ligne : Lien61

Blogs C

Nouvelle version v2.5.0 de la bibliothèque OCILIB

Une nouvelle version (2.5.0) de la bibliothèque OCILIB (C Wrapper for Oracle OCI) est disponible.
OCILIB est une librairie Open Source multi plateformes, écrite en C, proposant une gamme d'APIs permettant d'accéder très facilement à des bases de données Oracle et d'exécuter des ordres SQL et PL/SQL.
Contenu de cette version **2.5.0** :

- Ajout du support des PL/SQL tables : les tableaux PL/SQL peuvent être dorénavant liés (en input et output) à des tableaux natif C lors de l'appel à des procédures stockées
- Ajout de nouvelles fonctions de gestion de dates et timestamps
- Les objets OCI_Date, OCI_Timestamp et OCI_Interval peuvent dorénavant être indépendants des objets OCI_Connection
- Ajout de nouveaux attributs aux objets OCI_Connection
- Ajout de la possibilité de récupérer l'output du serveur généré par les procédures stockées
- Ajout d'un nouveau type de données : OCI_Threadkey permettant de gérer des variables identifiables par un littéral et ayant des valeurs spécifiques à chaque thread
- Extension de la gestion du chargement dynamique des librairies partagées sur les plateformes Unix-like
- Ajout du support de l'Instant Client sur les plateformes Unix-like
- Nouvelles validation de plateformes : Mac OS X et OpenVMS
- Ajout 26 sources complets d'exemples d'utilisation de OCILIB
- Documentation : mise à jour et ajout section "Installation"

Télécharger OCILIB **v2.5.0** (Lien62)
ChangeLog complet de la version 2.5.0 (Lien63)
Plateformes validées : Windows, Linux, HP/UX, Solaris, AIX, Mac OS X, OpenVMS
Compatibilités Oracle : 8i, 9i, 10g, 11g
Liens OCILIB Francophones :

- Site de OCILIB sur developpez.com (Lien64)

Liens OCILIB anglophones :

- Site (anglophone) de OCILIB sur sourceforge.net (Lien65)
- Blog de OCILIB (Lien66)

Retrouvez ce billet sur le blog d'Oracle : Lien67

Les derniers tutoriels et articles

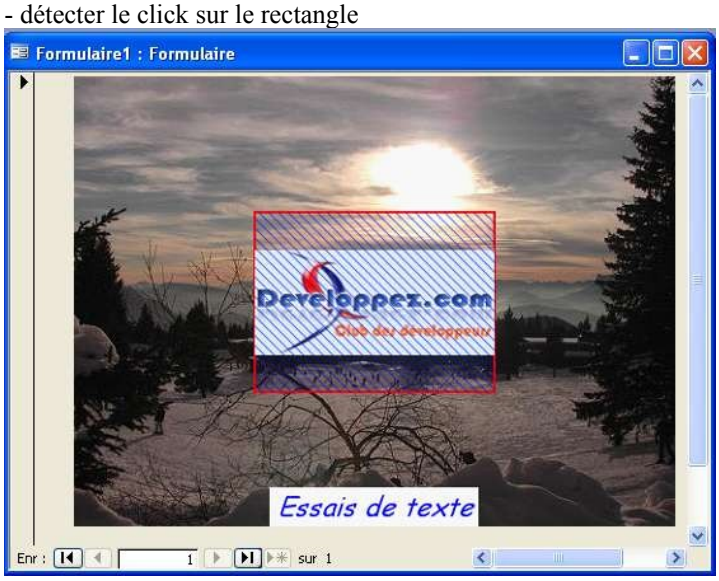
clGdiplus : Premiers pas

Ce document a pour objectif d'expliquer les bases de l'utilisation de la classe ClGdiPlus.
La classe ClGdiPlus apporte des fonctions pour dessiner sur un contrôle image standard et permet de rendre l'image interactive par l'ajout de régions sensibles aux actions de la souris.

1. Introduction

Ce tutoriel a été rédigé avec Access 2003.

Nous allons au cours de ce tutoriel :
- charger une image de fond
- dessiner du texte
- dessiner un rectangle
- dessiner une image dans ce rectangle par dessus l'image de fond
- détecter le survol du rectangle
- détecter le click sur le rectangle



Voici le résultat que l'on obtiendra à la fin du tutoriel

Consultez la documentation des fonctions et des propriétés (Lien68)

Lien vers la librairie en téléchargement sur Microsoft.com pour Windows autre que XP (Lien69)

2. Création du formulaire et du contrôle image

Créez un formulaire et placez-y un **contrôle image** de n'importe quelle taille.
Il n'est pas utile d'intégrer une image dans le contrôle car on va dessiner dessus.
Si nécessaire, choisissez une image quelconque pour créer le contrôle puis dans les propriétés du contrôle dans l'onglet **Format**, effacez la propriété **Image**.
Définissez le **mode d'affichage à Zoom**. L'image sera redimensionnée en conservant ses proportions.
Vérifiez le **nom du contrôle** dans l'onglet **Autres**, changez le si-besoin en **Image0** (c'est le nom qu'on va utiliser dans ce tutoriel)

3. Création de la classe

Télécharger la classe au format texte (Lien70)
Créez un **nouveau module de classe**.
Collez-y le contenu du fichier texte.
Sauvegardez le module avec le nom **ClGdiPlus**.
Attention : le nom sous lequel vous sauvegardez le module est important.

Ce module de classe VBA utilise la librairie gdiplus.dll de Microsoft.
Si vous utilisez une version de Windows autre que XP, télécharger la librairie et placez la dans le même répertoire que le fichier Access.

4. Déclaration de la classe et initialisation

On va écrire notre code dans le module du formulaire. Cliquez sur **Affichage** --> **Code** pour ouvrir ce module.
Vérifiez que vous avez l'instruction **Option Explicit** en haut du module.
Sinon rajoutez le pour imposer la déclaration de toutes les variables, cela évite les étourderies.

En-tête de module

```
Option Compare Database
Option Explicit
```

Pour pouvoir utiliser la classe il est nécessaire de la déclarer.
La classe est un objet dont le type est le nom sous lequel on a sauvegardé notre module de classe.
Elle se déclare comme n'importe quelle autre variable.

Déclaration de la classe

```
Private gGdip As clGdiPlus
```

Première chose à faire : il faut **créer une instance de l'objet classe**.
(Pour l'instant gGdip a pour valeur **Nothing**).

On écrit ce code dans l'événement **Sur Chargement** du formulaire.
Dans les propriétés du formulaire, définissez **[Procédure événementielle]** dans l'événement **Sur Chargement**.
Cliquez sur les trois petits points **[...]** pour générer l'événement dans le code.

Initialisation du contrôle

```
Private Sub Form_Load()
' Initialisation de la classe de dessin
Set gGdip = New clGdiPlus
End Sub
```

Deuxième chose à faire : **pensez à libérer la classe dès qu'elle**

n'est plus utile, au plus tard donc à la fermeture du formulaire.
La libération de la classe est importante car elle supprime tous les objets graphiques de la mémoire.

Dans les propriétés du formulaire, définissez **[Procédure événementielle]** dans l'événement **Sur Fermeture**.
Cliquez sur les trois petits points **[...]** pour générer l'événement dans le code.

A l'intérieur de la procédure **Form_Close** on va **libérer la classe** : il suffit de lui donner la valeur **Nothing** si elle n'a pas déjà cette valeur.

Libération de la classe
```
Private Sub Form_Close()
' Libération de la classe à la fermeture du formulaire
If Not gGdip is Nothing Then Set gGdip = Nothing
End Sub
```

5. Chargement d'une image de fond

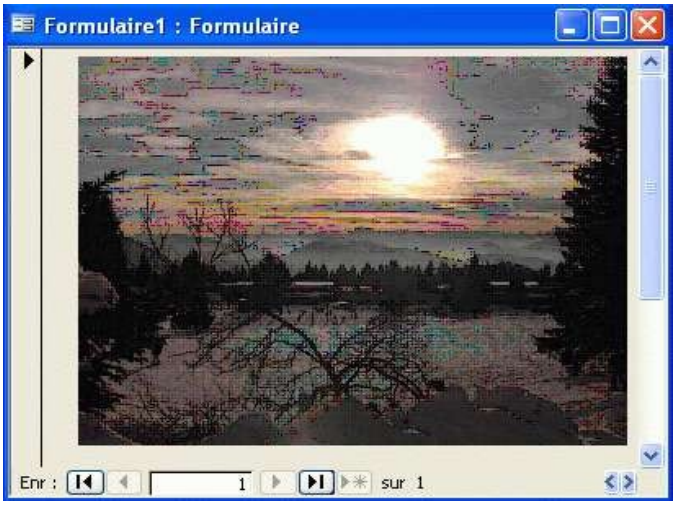
Si vous visualisez le formulaire, vous ne voyez rien.
C'est normal car on n'a encore rien fait...

On va maintenant **charger une image** dans le contrôle.
Placez une image dans le même répertoire que votre base de données.
Mon image s'appelle *DSCN1099.JPG*, remplacez ce nom de fichier par le vôtre.

La fonction utilisée pour charger le fichier est **OpenFile**.
Une fois le fichier ouvert, il faut ensuite mettre à jour le contrôle pour voir le résultat à l'écran.
(La fonction OpenFile ouvre le fichier en mémoire mais n'affiche rien).
L'affichage à l'écran s'effectue en mettant à jour la propriété **PictureData** de l'image grâce à la fonction **RepaintControl**.

Chargement d'une image de fond
```
Private Sub Form_Load()
' Initialisation de la classe de dessin
Set gGdip = New clGdiPlus
' Chargement d'une image de fond
gGdip.OpenFile CurrentProject.Path & "\DSCN1099.JPG"
' Affichage de l'image dans le contrôle
gGdip.RepaintControl Me.Image0
End Sub
```

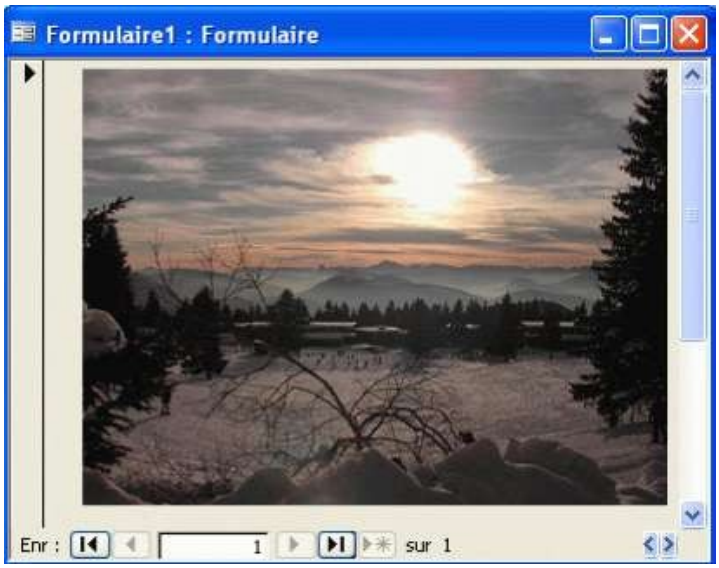
On obtient alors l'image dans le formulaire.



Il peut parfois y avoir un problème d'affichage au redimensionnement de l'image, généralement lors de l'affichage d'une image plus grande que le contrôle qui la contient.

Pour résoudre ce problème, passez le paramètre **pUseEMF** de la fonction **RepaintControl** à **Vrai**.
Le problème d'affichage ne se produit pas pour un type d'image EMF.

Affichage avec format EMF
```
' Affichage de l'image dans le contrôle
gGdip.RepaintControl Me.Image0, , , True
```



6. Dessin du texte

Nous souhaitons maintenant dessiner du texte sur l'image :
- Texte : Essais de texte
- Taille : 16 points
- Police : Comic sans MS
- Position : En bas, centré horizontalement
- Couleur du texte : Bleu
- Couleur du fond : Jaune pâle
- Italique : Oui
- Souligné : Non
- Barré : Non

- On passe 4 coordonnées en paramètres, ce sont les coordonnées d'un rectangle dans lequel on va dessiner le texte.
On utilise les coordonnées (**0,0,gGdip.ImageWidth, gGdip.ImageHeight**) qui sont celles de l'image complète.

- **1** et **2** sont les alignements du texte : centré horizontalement et positionné en bas dans le rectangle défini précédemment.

- **VbBlue** est la couleur du texte : on écrit le texte en bleu.

- **RGB(255, 255, 200)** correspond à une couleur de fond jaune pastel.

- **True** dans le paramètre pItalic pour afficher le texte en italique.

La taille du texte pour la fonction **DrawText** est exprimée en pixel sur l'image.
Pour l'exemple nous souhaitons que le texte affiché à l'écran soit de la même taille qu'un contrôle de taille de police égal à 16.

Cette valeur 16 est exprimée en points.

La fonction **FontSizeToPixel** va convertir les twips (ou points) en pixels sur l'image.

Pour tester j'ai placé un contrôle étiquette pour vérifier que la taille du texte correspond.

Dessine le texte
```
' Dessin du texte
gGdip.DrawText "Essais de texte",
gGdip.FontSizeToPixel(16 , Me.Image0), "Comic sans MS",
0, 0, gGdip.ImageWidth, gGdip.ImageHeight, 1, 2,
vbBlue, , RGB(255, 255, 500), , True
```

Placez ce code avant l'instruction d'affichage dans le contrôle.
Une seule mise à jour de la propriété **PictureData** du contrôle sera nécessaire une fois tout le dessin effectué en mémoire.

Voilà ce que l'on obtient.

7. Dessin du rectangle

On va **afficher un rectangle au centre de l'image**.

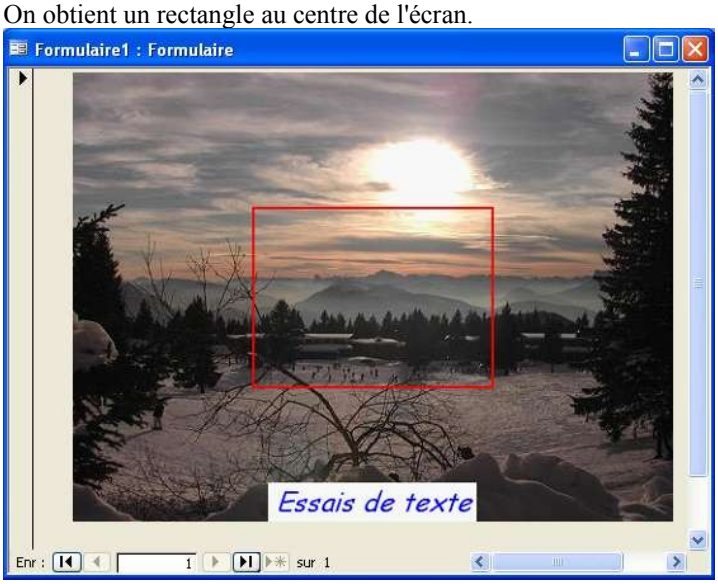
On donne les coordonnées du rectangle à dessiner à la fonction **DrawRectangle**.
On dessine ici un rectangle rouge dont la taille est égale à 2/5 de la taille de l'image visible.

L'intérieur du rectangle est transparent; pour ajouter une couleur de remplissage, remplacez -1 par un code couleur.

Dessine un rectangle centré sur le contrôle
```
' Dessin du rectangle
gGdip.DrawRectangle gGdip.ImageWidth / 2 -
gGdip.ImageWidth / 5, _
                    gGdip.ImageHeight / 2 -
gGdip.ImageHeight / 5, _
                    gGdip.ImageWidth / 2 +
gGdip.ImageWidth / 5, _
                    gGdip.ImageHeight / 2 +
gGdip.ImageHeight / 5, _
                    , vbRed, 2
```

Placez ce code avant l'instruction d'affichage dans le contrôle.

On obtient un rectangle au centre de l'écran.



8. Affichage d'une petite image dans le rectangle

Maintenant on va afficher une petite image à l'intérieur du rectangle que l'on vient de dessiner.

On écrit toujours le code avant l'instruction d'affichage dans le contrôle.

Pour dessiner une image il faut d'abord l'**ajouter à la liste d'images**.

Comme pour le chargement de l'image de fond on défini le chemin du fichier et on limite la taille de l'image chargée à la largeur du rectangle qui va contenir l'image.

On donne un nom à cette image : **MonImage**.

Ajoute l'image à la liste d'images
```
' Ajout de l'image à la liste d'images
gGdip.ImageListAdd "MonImage", CurrentProject.Path & "\
logo.gif", 2 * gGdip.ImageWidth / 5
```

On va ensuite **dessiner l'image "MonImage"** dans le rectangle.

On utilise la fonction **DrawImage** qui permet de dessiner une image de la liste d'images.
On donne en paramètres de la fonction les mêmes coordonnées que pour le rectangle.
Puis on défini le **mode d'affichage** à **GdipSizeModezoom** et le **positionnement** à **GdipAlignCenter**.

L'image est dessinée dans le rectangle comme elle le serait dans un contrôle image avec les mêmes propriétés de mode d'affichage et de positionnement.

Si votre image a une couleur de transparence mettez cette couleur de transparence à la place du -1 (vbWhite par exemple).

Dessine une image dans le rectangle rouge
```
' Dessine l'image dans le rectangle
' Mode Zoom centré
gGdip.DrawImage "MonImage", gGdip.ImageWidth / 2 -
gGdip.ImageWidth / 5, _
       gGdip.ImageHeight / 2 - gGdip.ImageHeight / 5, _
       gGdip.ImageWidth / 2 + gGdip.ImageWidth / 5, _
       gGdip.ImageHeight / 2 + gGdip.ImageHeight / 5, _
       -1, GdipSizeModezoom, GdipAlignCenter
```

On n'a plus besoin de l'image donc on la supprime.

Supprime l'image de la liste d'images
```
' Supprime l'image de la liste
gGdip.ImageListDel "MonImage"
```

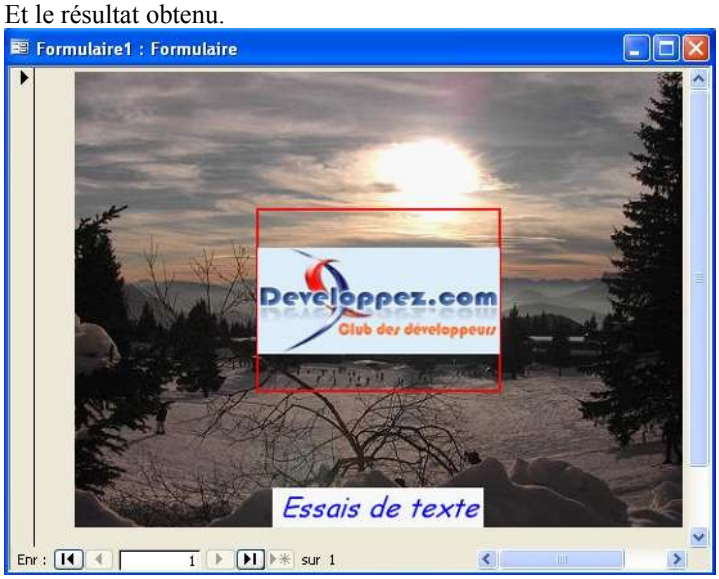
On a fini de dessiner!

Voilà le code complet qui s'exécute au chargement du formulaire.

Code de dessin
```
Option Compare Database
Option Explicit

' Déclaration de la classe
Private gGdip As New ClGdiPLus

Private Sub Form_Close()
' Libération de la classe à la fermeture du formulaire
If Not gGdip Is Nothing Then Set gGdip = Nothing
End Sub

Private Sub Form_Load()
' Initialisation de la classe de dessin
Set gGdip = New ClGdiPLus
' Chargement d'une image de fond
' On précise la largeur de l'image car il n'est pas
nécessaire de charger une image plus large que le
contrôle
' La hauteur sera automatiquement calculée en fonction
```

```
de la largeur précisée et des proportions de l'image
gGdip.OpenFile CurrentProject.Path & "\DSCN1099.JPG"
' Dessin du texte
gGdip.DrawText "Essais de texte",
gGdip.FontSizeToPixel(16 , Me.Image0), "Comic sans MS",
0, 0, gGdip.ImageWidth, gGdip.ImageHeight, 1, 2,
vbBlue, , RGB(255, 255, 500), , True
' Dessin du rectangle
gGdip.DrawRectangle gGdip.ImageWidth / 2 -
gGdip.ImageWidth / 5, _
                    gGdip.ImageHeight / 2 -
gGdip.ImageHeight / 5, _
                    gGdip.ImageWidth / 2 +
gGdip.ImageWidth / 5, _
                    gGdip.ImageHeight / 2 +
gGdip.ImageHeight / 5, _
                    , vbRed, 2
' Ajout de l'image à la liste d'images
gGdip.ImageListAdd "MonImage", CurrentProject.Path & "\
logo.gif", 2 * gGdip.ImageWidth / 5
' Dessine l'image dans le rectangle
' Mode Zoom centré
gGdip.DrawImage "MonImage", gGdip.ImageWidth / 2 -
gGdip.ImageWidth / 5, _
                    gGdip.ImageHeight / 2 -
gGdip.ImageHeight / 5, _
                    gGdip.ImageWidth / 2 +
gGdip.ImageWidth / 5, _
                    gGdip.ImageHeight / 2 +
gGdip.ImageHeight / 5, _
                    -1, GdipSizeModezoom,
GdipAlignCenter
                    gGdip.ImageListGetPixel "MonImage",
1, 1
' Supprime l'image de la liste
gGdip.ImageListDel "MonImage"
' Affichage de l'image dans le contrôle
gGdip.RepaintControl Me.Image0, , , True
End Sub
```

Et le résultat obtenu.



On peut à présent passer à l'interactivité...

9.La définition d'une région

Dessiner sur un contrôle c'est bien mais c'est encore mieux de rendre des zones sensibles et de réagir aux actions de l'utilisateur.

Pour faire ça il faut passer par la création de régions.

Pour créer une région on peut utiliser les fonctions dédiées : **CreateRegionRect**,**CreateRegionPolygon** et **CreateRegionEllipse**.

Pour créer une région correspondant au rectangle rouge cela donne :

Crée une région correspondant au rectangle rouge

```
gGdip.CreateRegionRect "MaRegion", gGdip.ImageWidth / 2
- gGdip.ImageWidth / 5, _
        gGdip.ImageHeight / 2 - gGdip.ImageHeight / 5, _
        gGdip.ImageWidth / 2 + gGdip.ImageWidth / 5, _
        gGdip.ImageHeight / 2 + gGdip.ImageHeight / 5
```

On voit que l'on serait obligé de passer à nouveau les mêmes coordonnées en paramètres.
Il y a plus simple, il suffit d'utiliser le paramètre **pRegion** de la fonction **DrawRectangle** avec laquelle on a dessiné le rectangle.

On modifie alors le précédent appel à la fonction **DrawRectangle** comme suit :

Crée une région correspondant au rectangle rouge

```
' Dessin du rectangle
gGdip.DrawRectangle gGdip.ImageWidth / 2 -
gGdip.ImageWidth / 5, _
        gGdip.ImageHeight / 2 - gGdip.ImageHeight / 5, _
        gGdip.ImageWidth / 2 + gGdip.ImageWidth / 5, _
        gGdip.ImageHeight / 2 + gGdip.ImageHeight / 5, _
        , vbRed, 2, , , "MaRegion"
```

On a simplement rajouté le paramètre **"MaRegion"** à la fin.
Cela va **créer automatiquement une région** nommée **"MaRegion"** définie par les coordonnées du rectangle dessiné.

10. Détecter le clic sur une région

On voudrait dans la suite afficher une boîte de message si l'utilisateur clique dans le rectangle rouge.

L'évenement **Sur clic** du contrôle image ne propose pas de paramètre donnant la position de la souris alors on va utiliser l'événement **Sur souris appuyée**.

La fonction **GetRegionXY** nous renvoie le nom de la région située sous le curseur de la souris.
Notez qu'on converti d'abord la position de la souris en pixel sur l'image avec les fonctions **CtrlToImgX** et **CtrlToImgY**.

Si on a cliqué dans le rectangle rouge on reçoit alors la valeur **"MaRegion"**.
On peut à ce moment afficher une boîte de message.

Affiche un message si on clique dans le rectangle rouge

```
Private Sub Image0_MouseDown(Button As Integer, Shift
As Integer, X As Single, Y As Single)
Dim lRegion As String
' On teste si la classe est initialisée
If gGdip Is Nothing Then Exit Sub
' Récupère le nom de la région sur laquelle on a cliqué
lRegion = gGdip.GetRegionXY(gGdip.CtrlToImgX(X,
Me.Image0), gGdip.CtrlToImgY(Y, Me.Image0))
' Si on a cliqué dans le rectangle rouge alors on
affiche un message
If lRegion = "MaRegion" Then MsgBox "Vous avez cliqué
dans le rectangle rouge!"
End Sub
```

On obtient bien un message seulement si on clique dans le rectangle.

Mais on aimerait faire mieux parce qu'on n'a pas d'information qui nous indique qu'on survole une zone sensible de l'image...

11. Détecter le survol d'une région

On va, pour terminer, déterminer si on survole notre rectangle et

le hachurer.

Pour savoir le nom de la région survolée on va procéder de la même manière que pour le clic mais sur l'événement **Sur souris déplacée**.

Si on survole le rectangle on souhaite hachurer la région avec la fonction **HatchRegion**.

On utilise les fonctions de sauvegarde en mémoire (**KeepImage** et **ResetImage**) pour rétablir l'image avant de la hachurer.
Ainsi on repart à chaque fois de l'image d'origine.
Il faut donc exécuter la fonction **KeepImage** une fois l'image dessinée dans la procédure **Form_Load**

La variable Static **sRegion** nous est utile pour ne dessiner ou retirer les hachures que si la région survolée change.

Ajout dans Form_Load à la fin de la procédure

```
' Conserve l'image
gGdip.KeepImage
```

Hachure la région lorsqu'on survole le rectangle

```
Private Sub Image0_MouseMove(Button As Integer, Shift
As Integer, X As Single, Y As Single)
Dim lRegion As String
Static sRegion As String
' On teste si la classe est initialisée
If gGdip Is Nothing Then Exit Sub
' Récupère le nom de la région sur laquelle on a cliqué
lRegion = gGdip.GetRegionXY(gGdip.CtrlToImgX(X,
Me.Image0), gGdip.CtrlToImgY(Y, Me.Image0))
' si changement de région
If sRegion <> lRegion Then
    ' Rétabli l'image originale
    gGdip.ResetImage
    ' Rempli la région en bleu
    gGdip.HatchRegion lRegion, vbBlue, ,
HatchStyleForwardDiagonal, 150
    ' Affichage de l'image dans le contrôle
    gGdip.RepaintControl Me.Image0, , , True
End If
' Conserve le nom de la dernière région survolée
sRegion = lRegion
End Sub
```

Voilà c'est fini on a bien notre région hachurée au survol du rectangle.

12. Corriger les clignotements

Le changement d'image lors du survol de la souris peut produire un effet de clignotement.
Il semble que le couple Access 2003 + Windows XP soit le plus touché par cet effet.

Trois solutions pour essayer de se débarrasser de ces clignotements :

- Utiliser la fonction **SetXPTheme** pour désactiver le thème XP sur les contrôles de l'application.
Désactiver le thème XP pour faire disparaître les scintillements.

- Utiliser la fonction **SetDoubleBufferXP** pour activer le double tampon sur le formulaire.
Le double tampon élimine efficacement les scintillements de l'image, mais est gourmand en ressource système.

- Utiliser la fonction **FastRepaint** pour redessiner l'image directement sur le formulaire.
Au lieu de modifier la propriété **PictureData** du contrôle, dessiner directement l'image sur le formulaire.
Cette fonction dessine temporairement l'image.

13. Conclusion

On a donc dessiné sur un contrôle image et on a rendu une zone sensible pour donner de l'interactivité à l'image.

Retrouvez les articles sur clGdiplus de Thierry Gasperment en ligne : Lien71

Création et manipulation d'un planning - Partie 1

Cet article se propose de démonter le mécanisme de création d'un planning.

1. Introduction

Comment gérer un planning sous Access ?...
Question souvent posée sur le forum. Je me propose donc au travers de la base de données "GESTION DE PLANNING" (téléchargeable) de démonter le mécanisme de création d'un planning.
Dans cette base, j'ai souhaité montrer un maximum d'éléments.
Quelques exemples :

- comment personnaliser les boutons et gérer les images (voir Caféine (Lien72))
- comment appeler la palette des couleurs (Appeler une API – voir Tofalu (Lien73))
- comment changer le pointeur de la souris (Appeler une API – voir Faw (Lien74))
- comment enrichir le contenu d'une liste déroulante (voir Maxence Hubiche (Lien75))
- comment alimenter une liste déroulante en fonction d'une autre (voir Jean-Philippe Ambrosino (Lien76))
- manipuler les infos bulles (voir Morgan Billy (Lien77))
- listes déroulantes imbriquées, évènement sur déplacement de souris et autres, etc...

Cet article se divisera en deux parties.

1ère partie : Conception/structure de la base et Création du planning

Conception/structure de la base

1. Cahier des Charges
2. Structure des tables
3. Modèle relationnel

Création du planning (Formulaires nécessaires)

1. F_SaisieSalles (permet de saisir une nouvelle salle de formation)
2. F_SaisieCatalogue (permet de saisir le catalogue de formation)
3. F_SaisieFormateur (permet de saisir le profil d'un formateur)
4. F_Reservation (permet de créer une réservation)
5. F_Planning (Plate forme de toute l'application - affiche le planning et permet de générer une réservation, saisir une salle, un formateur, un nouvel item du catalogue)

2eme partie : Manipulations du planning, impression et envoi du planning des formateurs par mail

Manipulations du planning

1. Changer la date d'une formation

2. Changer de formateur
3. Changer de salle de formation
4. Supprimer une formation
5. Basculer l'affichage Salle de formation / Formateurs

Communiquer le planning

1. Imprimer le planning
2. Envoyer le planning du formateur par mail

1) Nous ne traiterons pas de la gestion des barres de menus, vous trouverez toutes les informations nécessaires sur le tuto "Personnalisez vos barres de commandes dans Access" de Starec (Lien78).
2) Bien que le code VBA soit omniprésent dans l'application, je m'efforcerai d'aborder et d'expliquer les choses simplement afin de mettre cet article à la portée du plus grand nombre d'utilisateurs.

2. Conception et analyse

2.1. Le Cahier des Charges

Objet du projet :
Décrire le process de génération d'un planning sur Access.
Base de travail :
Suivre le planning de l'occupation des salles et des formateurs d'un centre de formation.
La saisie des formations se fait directement à partir du planning par clic sur une plage de celui-ci.
De manière à revenir sur l'imbrication des listes déroulantes et des liens entre listes (déroulantes ou non) avec sous formulaire, la saisie d'une formation se fera à partir de la famille du produit (liste déroulante) qui de ce fait affichera les produits concernés (Liste déroulante dépendante).
Les formations du Catalogue liées à la famille de produit et au produit s'afficheront dans un sous formulaire.
Par clic sur la formation choisie, on rafraichira la liste des formateurs compétents disponibles pour la formation.
Un résumé de la formation apparaitra dans une info-bulle sur le pavé correspondant du planning.
Dans l'idée de voir évoluer notre centre de formation, il sera possible d'ajouter de nouvelles salles. Cependant, les numéros devront rester séquentiels sans rupture de séquence.
D'autre part, une formation ne pourra être tronquée par un weekend.

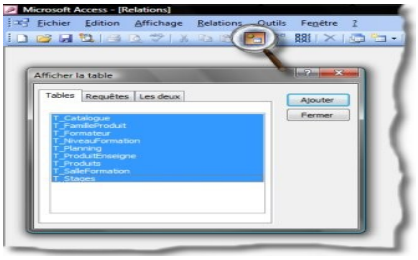
2.2. Les Tables nécessaires

Je n'ai repris dans les tables que les champs strictement nécessaires à l'application

(Consulter l'article sur le site pour voir les structures des tables (Lien79))

2.3. Le Modèle relationnel

Une fois les tables établies, il est maintenant temps de mettre en place les relations entre celles-ci.
Pour mettre en place les relations, il faut ouvrir la fenêtre des relations en cliquant sur l'outil "Relations" dans la barre d'outils de la fenêtre "Base de données".



Cliquez sur l'outil "Ajouter une table" et choisir dans la boite de dialogue les tables concernées par l'opération.

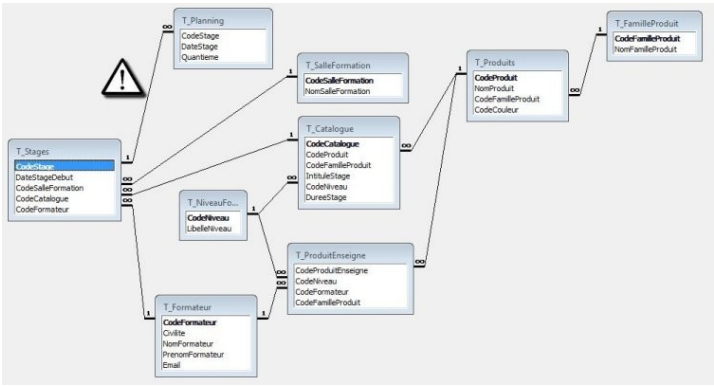
Pour créer une relation, il suffit de cliquer maintenu sur le champ de la table MERE et de glisser sur le champ à mettre en relation dans la table FILLE.
En lâchant la souris, Access affiche la boite de dialogue ci-contre.

1 - Access affiche le type de relation.

2 - Cocher cette option pour activer l'intégrité référentielle. Cette option a pour effet de contrôler la cohérence des données. Cela signifie :
qu'il sera impossible de supprimer une enregistrement PERE si des enregistrements FILS sont attachés.
exemple : Je ne pourrai pas supprimer un formateur si des stages lui sont attribués.
De même, je ne pourrai pas créer un enregistrement FILS si l'enregistrement PERE, de la table liée, n'existe pas.
exemple : Je ne pourrai pas créer un nouveau produit si la famille de ce produit n'existe pas.

3 - En activant cette option, Access supprimera tous les enregistrements des tables filles attachés à l'enregistrement PERE supprimé.
exemple : Nous activerons cette option sur la relation qui existe entre T_Stages et T_Planning. Ainsi la suppression d'un stage entrainera la suppression de toutes les lignes de la table T_Planning attachées à ce stage.

Lorsque toutes les relations ont été créées, nous obtenons l'écran suivant



3. Les Formulaires

3.1. Principes des boutons personnalisés

Il est possible de créer ses propres boutons et de leur donner un aspect dynamique.
Cette section va présenter ce processus. Le même processus est utilisé sur tous les boutons dessinés dans les différents formulaires

3.1.1. Description des boutons

Pour pouvoir donner l'effet d'animation, 3 images seront

nécessaires.

N°	Nom du bouton	Commentaire
1	btnFermerR	**L'effet relâché**. Visible par défaut, sera la première image de la pile
2	btnFermerS	**L'effet survolé**. Placée sous l'image 1, sera visible lors du passage de la souris sur le "bouton relaché"
3	btnFermerC	**L'effet cliqué**. Placée sous l'image 2, sera rendu visible lorsque l'on appuie sur le bouton de la souris.

3.1.2. Mécanisme de l'animation

Pour donner l'illusion d'un clic, il faut décomposer le mouvement :

1. Je survole le bouton qui est au premier plan.
   Celui-ci doit donc s'effacer pour montrer le bouton effet survolé. Cet effet permet de donner l'illusion à l'utilisateur que le bouton est sélectionné.
   On agira donc sur l'évènement "sur souris déplacée"
2. Je clique sur le bouton mis en évidence.
   Celui-ci doit donc s'effacer pour afficher le bouton effet cliqué.
   On agira donc sur l'évènement "sur souris appuyée".
3. Je lâche la souris. Le geste du clic sera donc complet. En relâchant la souris, le bouton effet cliqué doit s'effacer pour remontrer l'effet survolé puisque ma souris est toujours au dessus du bouton.
   On agira donc sur l'évènement "sur souris relâchée".

Cependant, lorsque la souris ne survole pas de bouton, il faut que l'effet relâché revienne au premier plan. Pour donner cet effet, on agira sur l'évènement "souris relâchée" de la section du formulaire où se trouve le bouton.

3.1.3. Le Code VBA

3.1.3.1 Chargement des images

Les images sont toutes stockées dans un dossier "image". Celles-ci sont chargées à l'ouverture de chaque formulaire.
Exemple de chargement des images du formulaire : F_SaisieSalle

```
Private Sub Form_Open(Cancel As Integer)
    ' Chargement des images
    btnFermerR.Picture = CurrentProject.Path & "\image\
btnFermerR.jpg"
    btnFermerS.Picture = CurrentProject.Path & "\image\
btnFermerS.jpg"
    btnFermerC.Picture = CurrentProject.Path & "\image\
btnFermerC.jpg"
    btnNouveauR.Picture = CurrentProject.Path &
"\image\btnNouveauR.jpg"
    btnNouveauS.Picture = CurrentProject.Path &
"\image\btnNouveauS.jpg"
    btnNouveauC.Picture = CurrentProject.Path &
"\image\btnNouveauC.jpg"
End Sub
```

Pour plus d'informations sur la gestion d'images avec Access, lire le tuto de Caféine (Lien72)

Ce code sera placé sur l'évènement "sur ouverture" du formulaire.
Le chemin de stockage de l'image est donné par la concaténation du chemin de l'application et du nom du fichier image à récupérer.
Ce code est bien sûr fonction du nombre de boutons à charger dans le formulaire.

3.1.3.2. Le code VBA pour les évènements de type "Déplacé" et "Appuyé"

Pour les évènements "sur souris déplacée" et "sur souris appuyée", on utilisera une fonction qui réagira en rapport avec les images à afficher ou à masquer.
Descriptif de la fonction :

```
Function BasculerBouton(CtlMasque As String, CtlAffiche
As String)
' frmActif représente le formulaire en cours
d'utilisation
    frmActif.Controls(CtlAffiche).Visible = True
    frmActif.Controls(CtlMasque).Visible = False
End Function
```

Au niveau des arguments, on indiquera le nom des boutons concernés par l'opération de bascule.
On remarque dans le code : "**frmActif**".
Il s'agit là d'une variable publique déclarée, dans un module spécifique, par :

```
' Variable pour récupération du formulaire en cours
d'utilisation pour animation des boutons
Public frmActif As Form
```

L'implantation de la fonction sur les boutons se fera via la fenêtre des Propriétés.

Retrouvez la suite de l'article de Jean Ballat en ligne : Lien79

Exécution d'applications X à distance

Cet article a pour but de vous présenter l'exécution d'applications X Window à distance

1. Introduction

X Window , souvent abrégé X11 ou encore X est le système de fenêtrage sur tous les systèmes Unix et dérivés. Ce système est très puissant car il est totalement indépendant du système d'exploitation sur lequel il tourne ou encore du matériel utilisé par ce système.

2. Théorie

La plus grande caractéristique du système X window est son architecture client/serveur qui permet une totale séparation entre les programmes qui gèrent le matériel et ceux qui doivent afficher des choses à l'écran. Ainsi le client et le serveur peuvent être sur des machines totalement différentes et très éloignées sans que cela ne gène.

C'est le serveur X qui se charge de la gestion du matériel. Actuellement la principale implémentation de X est Xorg, fork de Xfree86. La gestion du matériel ainsi que des unités d'affichage se fait via le fichier xorg.conf situé dans le répertoire /etc/X11/.

Le matériel se décompose en périphériques d'entrée (InputDevice) avec le clavier et la souris, en moniteurs qui représentent les écrans connectés aux différentes cartes graphiques (Device) présent sur la machine. Le regroupement d'un moniteur avec une carte forme un écran. Chaque serveur X de lancé forme ce que l'on appelle une unité d'affichage ou encore DISPLAY.

Chaque unité d'affichage est caractérisée par un identifiant de la forme nom_machine:numero_sequence ou nom_machine/unix:numero_sequence avec nom_machine qui est le nom de la machine sur laquelle elle tourne. Le numéro de séquence est un numéro qui commence à toujours 0 et permet de repérer les différents serveurs X de lancés . Dans l'identifiant, le nom de la machine peut être enlevé signifiant ainsi que l'on est sur localhost. La présence de /unix sert en autre à garder une compatibilité avec les anciennes versions mais surtout à dire que l'on va écouter sur un socket unix (donc sur localhost) et non tcp.

Le client X est un simple logiciel graphique qui se connecte au serveur X pour lui envoyer ses requêtes d'affichage via le protocole X au travers de la bibliothèque X (Xlib).

3. Pratique

Dans toute cette partie le serveur aura comme nom david (adresse serv.ath.cx) serveur et le client izu (adresse cli.ath.cx).

3.1. Configuration du serveur Linux

Pour qu'une personne puise se connecter à un serveur X, il faut qu'ils partagent entre eux un secret, un magic-cookie pour être précis. En détail, un magic-cookie est une chaîne de 128 bits, soit 32 caractères en hexadécimal. N'importe quelle chaîne répondant à ce critère de longueur peut être utilisée en tant que cookie.

La liste des personnes autorisées à se connecter au serveur X est dans le fichier spécifié via l'option -auth de X. Ce fichier est écrit par le login manager (GDM,KDM,...), il varie donc en fonction de celui ci. La visualisation ou l'édition de ce fichier ne peut se faire que via le programme xauth en root. Pour spécifier le fichier que l'on va utiliser , il faut passer par l'option -f de xauth. Pour le moment regardons qui est autorisé à se connecter à X.

```
debian@debian:~$ xauth -f /var/lib/gdm/:0.Xauth list
#ffff##:0  MIT-MAGIC-COOKIE-1
43cacbf710e9d4763869bbaa0c17bfac
```

Donc si on décortique tout cela, #ffff## signifie que tout le monde peut se connecter. 0 est le numéro du serveur X. MIT-MAGIC-COOKIE-1 est le protocole d'autorisation utilisé et la longue chaîne est la cookie en question.

Le serveur X, comme tout bon serveur écoute sur un port. De façon générale, un serveur X avec comme numéro de séquence N va écouter sur le port 6000+N. En d'autres termes, le 1er serveur X va écouter sur le port 6000 (son numéro de séquence vaut 0), le 2eme sur le port 6001, ... Si vous vous situez derrière un routeur, il faut donc veiller à ce que ce dernier redirige bien le bon port vers le serveur. Pour cela, je vous renvoie au manuel de ce dernier.

Enfin, il faut aussi s'assurer que le serveur X n'a pas été lancé avec l'option -nolisten tcp, option qui empêche le serveur d'écouter sur les ports. Actuellement, cette option est souvent mise par défaut. Pour le vérifier, exécutons un ps aux | grep X

```
david@debian:~$ ps aux | grep X
root      2855  5.1 11.6  79996 60408 tty7     Ss+
20:13   0:40 /usr/bin/X :0 -audit 0 -auth
/var/lib/gdm/:0.Xauth -nolisten tcp vt7
```

On voit donc que Xorg à été lancé avec cette option. Pour contourner le problème, 2 options: soit lancer un 2eme serveur X soit enlever l'option du serveur X actuel. Pour la 1ere option, c'est assez simple, il suffit de lancer en root :

```
X :1 vt8
```

Ainsi on lance un 2eme serveur X avec comme numéro de séquence 1 sur le terminal virtuel numéro 8. Il sera accessible via les touches Ctrl-Alt-F8.

Pour la 2eme solution, elle est dépendante du *login manager*. Par exemple pour GNOME, il faut lancer en root gdmsetup, puis aller dans l'onglet sécurité, puis décocher la case "Refuser les connections TCP au serveur X". Sous KDE, il faut éditer le fichier /etc/kde3/kdm/kdmrc et retirer -nolisten tcp de la ligne ServerArgsLocal=-nolisten tcp.

3.2. Configuration du client Linux

3.2.1. En console locale

La configuration est beaucoup plus simple que celle du serveur. Tout d'abord, on configure ~/.Xauthority via xauth. Pour cela, il suffit d'ajouter le serveur via la commande add. Son synopsis est le suivant : add identifiant typeprotocole cookie en veillant à utiliser le même cookie que celui du serveur. Ainsi, cela donne :

```
david@cli:~$ xauth add serv.ath.cx:0  MIT-MAGIC-COOKIE-1  43cacbf710e9d4763869bbaa0c17bfac
david@cli:~$ xauth add serv.ath.cx/unix:0  MIT-MAGIC-COOKIE-1  43cacbf710e9d4763869bbaa0c17bfac
```

En ensuite, il suffit de tester sur le client en passant à l'option display l'identifiant du serveur de la manière suivante:

```
xeyes -display=serv.ath.cx:0
```

Mais l'inconvénient de cette méthode, c'est que l'option display n'est pas supportée par tous les programmes. Des lors, une solution un peu plus contraignante mais plus universelle est de fixer la valeur de la variable d'environnement DISPLAY avec l'identifiant du serveur. Sous le shell bash, la commande est:

```
david@debian:~$ export DISPLAY=serv.ath.cx:0
```

Ensuite, il suffit de tester avec une application quelconque et si tout est bien configuré, vous devriez voir le résultat en quelque seconde.

3.2.2. Connexion sécurisée avec SSH

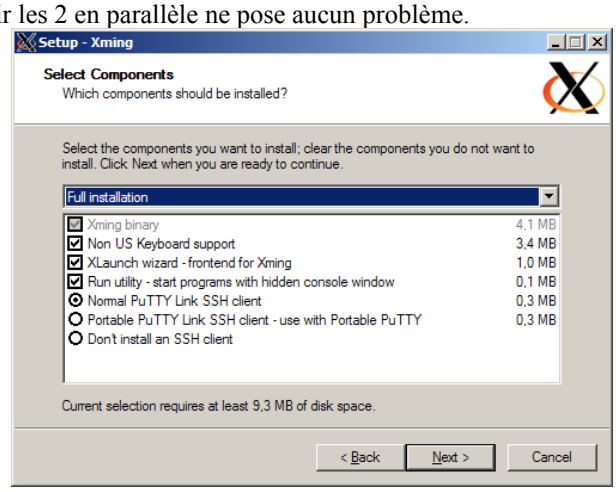
Attention, la méthode vue précédemment n'est à utiliser que dans un réseau sûr. En effet, le cookie va transiter en clair sur le réseau. Dès lors, il est très facile de le sniffer pour le récupérer. Si vous souhaitez utiliser cette technique sur internet, utilisez ssh pour transférer le cookie même si des lors la technique devient désuète avec l'option -X de ssh qui permet d'activer le *X11 forwarding*, c'est à d'activer la redirection de X11 qui permet alors de lancer de façon naturelle des applications X à distance. On peut aussi noter l'option -Y de ssh qui sert de joker à X quand celui ne marche pas en activant le *trusted X11 forwarding* (redirection d'X11 forcée). L'inconvénient avec cette dernière, c'est la sécurité qui est moindre puisque cette redirection ne va pas suivre les extensions de sécurité de X11.

Ainsi avec ssh, exécuter une application à distance revient à taper :

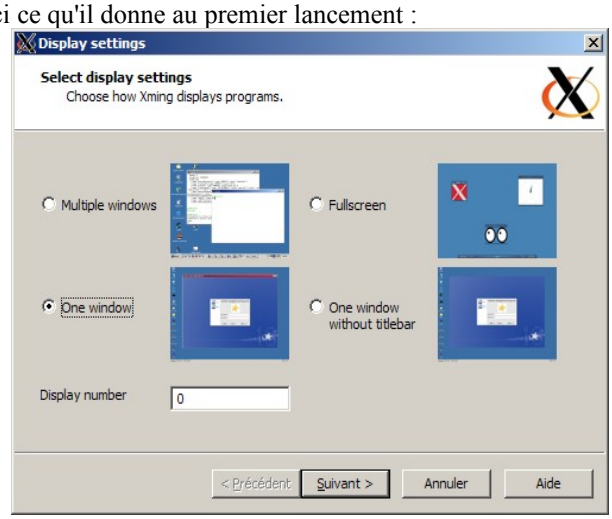
```
david@debian:~$ ssh -X test@cli.ath.cx
test@cli.ath.cx:~$ xeyes
```

3.3. Accéder au bureau linux depuis Windows

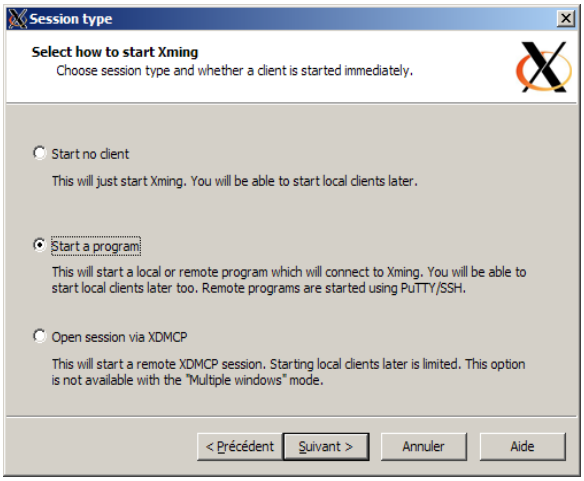
Le protocole d'accès au serveur X à distance transmet uniquement les instructions d'affichage des fenêtres à travers le réseau, dans le but d'optimiser les transferts de données qui sont ainsi très rapides. Mais cela a pour conséquence que la machine voulant afficher des applications X distantes doit également être équipée d'un serveur X local à même d'exécuter les instructions d'affichage. Il faudra donc émuler un serveur X sous Windows. Cygwin (Lien80) en propose un, mais c'est une solution lourde puisque cygwin est une émulation complète de l'environnement linux sous windows. Il existe un projet de serveur X sous Windows appelé **XMing** (Lien81), c'est cette solution que j'ai choisi ici. D'autre part, l'accès au serveur distant se fait via SSH. Il nous faut donc également un client SSH. Xming intègre un client SSH en utilisant le logiciel libre Putty (Lien82) (client ssh sous windows bien connu), mais vous pouvez choisir de ne pas l'installer si vous avez déjà Putty sur votre système. Néanmoins avoir les 2 en parallèle ne pose aucun problème.



Vous aurez ensuite 2 raccourcis : xlaunch et xming. Xming se contente de lancer le serveur X localement, tandis que xlaunch présente un assistant pour définir la façon dont nous désirons interagir avec le serveur X distant. C'est donc via xlaunch que nous ferons chaque fois la connexion.
Voici ce qu'il donne au premier lancement :

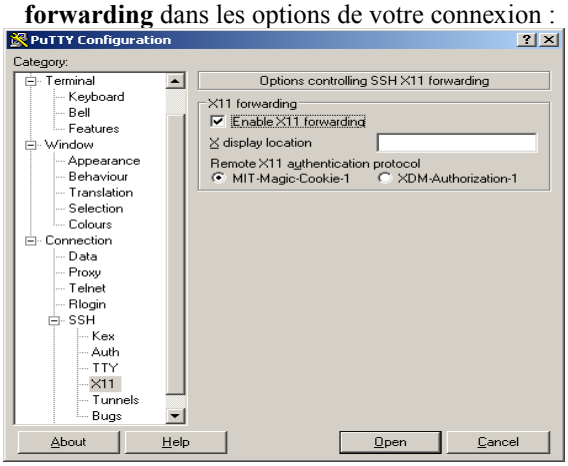


Vous pouvez donc choisir la façon dont les fenêtres seront affichées. En fait cela va dépendre également de la manière dont on se connecte au serveur X distant. Soit par SSH, à ce moment-là on n'a pas le bureau entier, mais on peut démarrer chaque application graphique une par une en ligne de commande, chaque application s'ouvrant à ce moment-là dans une fenêtre séparée. Soit en utilisant le protocole XDMCP qui permet d'ouvrir une session entièrement graphique qui reproduit tout le bureau X (équivalent du **Terminal Server** sous Windows). L'écran suivant permet d'ailleurs de sélectionner le mode souhaité :



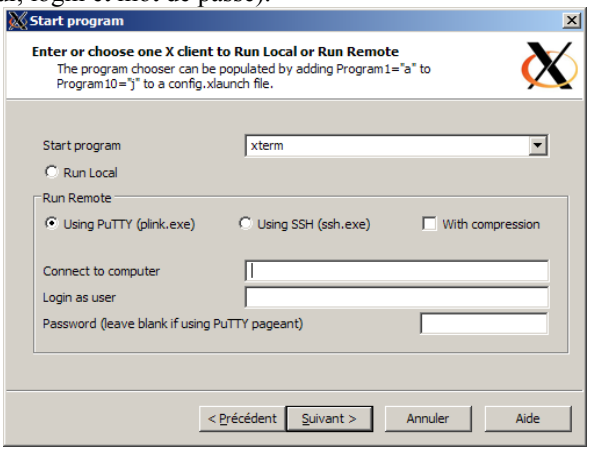
- **Start no client** : démarre le serveur X en mode démon seulement, nécessite de démarrer un client SSH soi-même (par exemple Putty) pour effectuer la connexion et lancer ensuite les programmes graphiques souhaités via

la console. Attention dans Putty il faut activer le **X11 forwarding** dans les options de votre connexion :

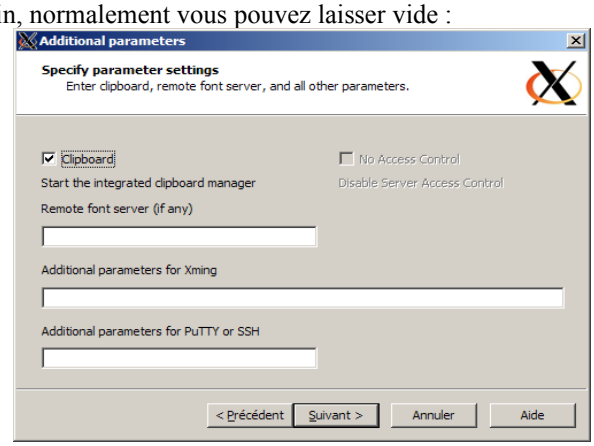


- **Start a program** : utilise le client SSH intégré de XMing pour effectuer la connexion distante au serveur X linux. Cela évite de lancer putty séparément.
- **Open session via XDMCP** : ici même le login sera graphique et géré par le serveur X distant; vous aurez en outre accès à l'entièreté du bureau comme si vous étiez physiquement devant la machine. C'est probablement le mode le plus pratique d'utilisation. Attention ne soyez pas surpris : dans ce mode le délai entre le login graphique et l'apparition du bureau est assez long (parfois plus d'une minute).

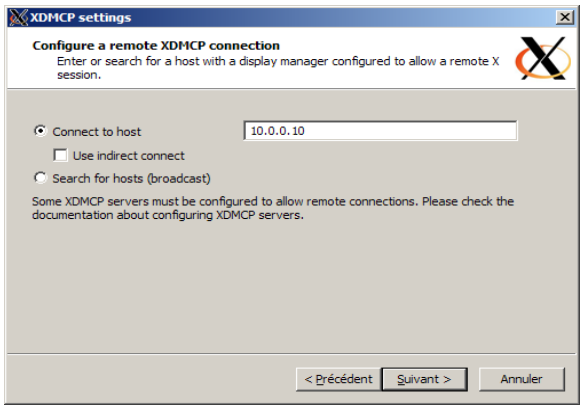
Si vous avez sélectionné le mode "start a program", XMing propose différents clients SSH, prenez l'option par défaut qui consiste à utiliser le client Putty intégré au package XMing. Vous devrez ici mentionner les informations de connexion (adresse du serveur, login et mot de passe).



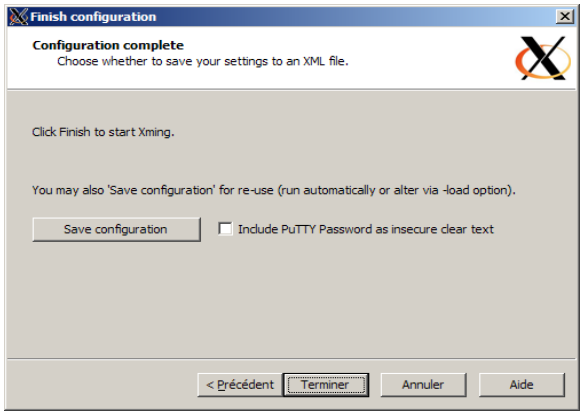
Vous pouvez spécifier des paramètres supplémentaires en cas de besoin, normalement vous pouvez laisser vide :



Dans le cas d'une connexion XDMCP, vous devrez seulement préciser l'adresse du serveur où se connecter puisque l'authentification se fera au travers du serveur X :

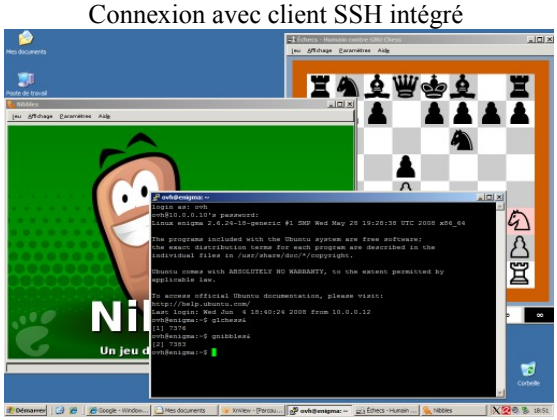


L'écran final permet de sauvegarder la configuration pour un prochain lancement et de démarrer le serveur X local :

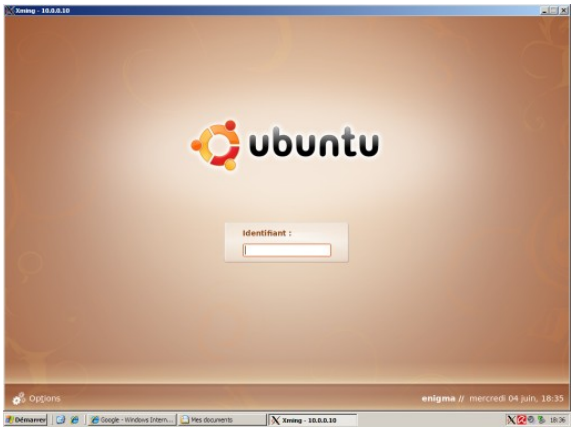


La configuration est maintenant terminée, voici les différents résultats suivants les modes de connexion :

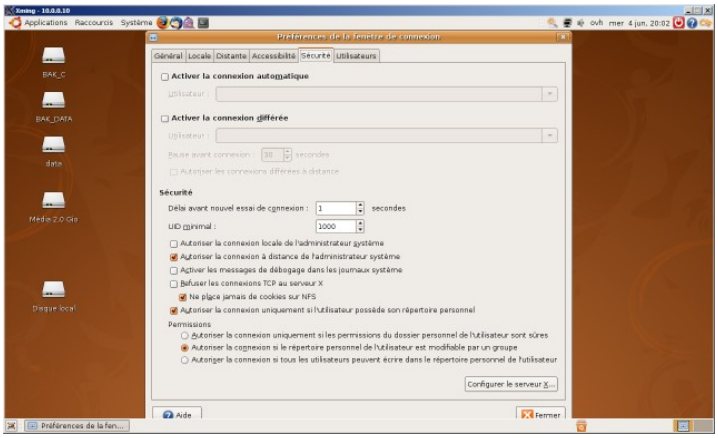
Connexion avec client SSH intégré



Connexion avec client Putty lancé à la main



Login graphique en XDMCP


Bureau complet en XDMCP, permet de voir au passage la configuration à effectuer pour le serveur X sous Ubuntu Linux 8.04.

4. Conclusion

J'espère qu'avec cet article, vous avez pris conscience de la puissance X ainsi que des applications que l'on peut en faire.

Retrouvez l'article David Come et Olivier Van Hoof en ligne : Lien83

Installation de XEN dans un environnement SAN

Cet article a pour but de décrire la mise en place de machines virtuelles sous XEN dans un environnement SAN. Il recense les différentes étapes nécessaires et les problèmes rencontrés lors du déploiement de nos machines virtuelles.

1. Introduction

Avant de rentrer dans le vif du sujet je vais faire un rappel sur les principales technologies de stockage passées et présentes et tenter de faire un état des lieux des nouvelles fonctionnalités apportées par la virtualisation et les différents types d'hyperviseurs.
Ensuite je donnerais quelques définitions que le lecteur se devra de connaître avant d'aborder l'installation de XEN dans un réseau de stockage de type SAN.
Bonne lecture.

1.1. Les principales technologies de stockage

1.1.1. SCSI

L'interface SCSI (*Small Computer System Interface*) est un bus permettant de gérer plusieurs périphériques.
Un seul bus SCSI peut accepter de 8 à 15 unités physiques (disques durs, scanners graphiques,lecteur de bande, etc...), sachant que la carte SCSI que vous utiliserez sera comptée parmi ces unités physiques.
Il y a eu 3 normes SCSI et parmi elles on trouve plusieurs déclinaisons d'interfaces (Wide SCSI, Fast SCSI, Ultra SCSI, Ultra 160 SCSI, Ultra 320 SCSI , Ultra 640 SCSI, etc...)
Pour plus d'informations sur cette technologie je vous conseille de visiter cette page (Lien84)

1.1.2. Le NAS (Network Attached System)

Le système de stockage NAS est comme son nom l'indique un système de stockage en réseau.
Sauf que contrairement au SAN, il n'utilise pas de réseau dédié au stockage, mais un réseau ethernet classique.
On parle aussi de **serveur NAS**, qui est un serveur dédié au stockage de données.
Accessible par le réseau il permettra aux autres machines d'y stocker leurs données.

1.1.3. Le SAN (Stockage Area Network)

Le SAN est un réseau dédié au stockage. Le type de média utilisé est la fibre optique.
On peut trouver sur un SAN différents éléments tels que des robots de sauvegarde, des baies de disques, des commutateurs (équivalents des commutateurs éthernet mais dédiés à la technologie SAN) sur lesquels les différents éléments du réseau vont venir se brancher.
Chaque élément d'un SAN est identifié par un WWN (World Wide Name).
Il s'agit d'une sorte de plaque d'immatriculation pour chaque périphérique un peu à la manière des MAC adresses pour les cartes réseau, sauf que le WWN est composé de 8 octets (6 octets pour les MAC adresses).

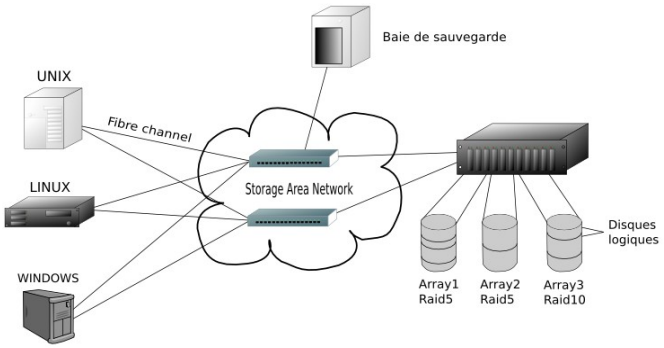
Les avantages du SAN sont les suivants:

- La fibre optique permet de très bons débits et la bande passante est uniquement dédiée aux opérations de lecture/écriture sur les différents éléments du réseau
- **La flexibilité:** On peut décider par une technique de zoning de faire voir ou de cacher différents éléments du réseau de stockage aux serveurs qui y sont reliés.
- **La flexibilité (le retour) :** Les possibilités d'évolution du réseau de stockage sont énormes. Tant que vous avez des ports sur votre commutateur SAN vous pouvez ajouter de la capacité de stockage.
- **Redondance:** Plusieurs cartes adaptatrices SAN permettront à un serveur d'avoir plusieurs chemins d'accès vers un ou plusieurs éléments du réseau de stockage. Un bon logiciel de multipathing et/ou de répartition de charge sauront gérer ça à merveille.
- **Partage de données:** Si plusieurs serveurs peuvent avoir accès au même contenu (technique de zoning) alors il nous sera facile de pouvoir monter des solutions de cluster haute disponibilité.

Les désavantages d'un SAN

- Technologie chère à mettre en oeuvre.

- Requière une formation de base avant de se lancer dans le déploiement d'un SAN.



1.2. Les hyperviseurs

Un hyperviseur est un système léger et adapté pour faire tourner des noyaux de systèmes invités.
On peut dire aussi des hyperviseurs qu'ils sont des minis OS légers conçus pour faire tourner plusieurs autres systèmes d'exploitation **en même temps sur une même plateforme matérielle.**

Aujourd'hui la virtualisation est une tendance de plus en plus à la mode voir même en plein boom.

L'intérêt de faire tourner plusieurs systèmes d'exploitation sur la même machine:

- **Réduction des coûts:** On évite de multiplier les machines physiques, on y gagne en place, en dépenses d'électricité et la puissance de calcul des dernières machines à la mode n'est plus sous exploitée.
- **Flexibilité:** Les derniers hyperviseurs ont des fonctionnalités intéressantes tel que l'ajout/retrait à chaud de ressources tel que la RAM, le stockage, le nombre de coeurs CPU, etc...
  Imaginez que vous avez trois machines virtuelles fonctionnant sur votre hyperviseur, et que vous savez qu'a une certaine période de la journée vous avez un pic d'activité sur la machine 1 alors que les deux autres machines ne sont pas beaucoup sollicitées.
  Vous pouvez alors choisir grâce à votre hyperviseur d'enlever de la puissance de calcul à vos deux machines non sollicitées pour l'ajouter à la machine 1 et tout ça à chaud bien sûr.
  Une fois le pic d'activité passé vous pouvez décider de répartir les ressources équitablement entre les trois machines.
- **Haute disponibilité/Redondance:** Bien sûr vous êtes en droit de vous demander, si ce n'est déjà fait, ce qui adviendrait de vos machines virtuelles si votre hyperviseur venait à tomber en panne ou encore plus simplement si vous deviez arrêter celui-ci pour maintenance. Et je vous répondrais immédiatement que vos machines virtuelles seraient autant indisponibles que le serait votre hyperviseur.
  Mais les formidables ingénieurs concepteurs d'hyperviseurs ont pensé à tout et ont ajouté à ceux ci une fonctionnalité qui permet à un hyperviseur A de confier la gestion de ses machines virtuelles à un hyperviseur B et tout cela une fois de plus sans interruption de services.
  Cependant un espace de stockage partagé est recommandé entre les deux hyperviseurs.

On s'abstrait des notions de serveur physique, aujourd'hui une machine virtuelle tourne sur un des hyperviseurs du système d'information. On a bien sûr moyen de savoir de quel hyperviseur il s'agit mais ce n'est pas essentiel tant que les services associés à cette machine virtuelle tournent.
Au cours de sa vie une machine virtuelle , pour peu que vous ayez programmé des maintenances automatiques sur vos hyperviseurs, sera amené à se déplacer d'un hyperviseur à un autre, mais une fois de plus tant qu'il n'y a pas d'interruption de services ce n'est pas un problème pour vous. A partir d'un seul poste vous pouvez avoir accès à de multiples informations sur vos serveurs virtuels (status du serveur, taux d'occupation de la ram, des cpus, etc...) et vous pouvez même agir sur ceux ci.
Pratiquement finis les déplacements en salle machine pour aller accéder à la console de la machine puisque vous y avez accès depuis votre poste.
Voilà ce qu'apporte la virtualisation et les hyperviseurs et j'en oublie certainement.

1.2.1. Différences entre les hyperviseurs de type 1 et de type 2

Les hyperviseurs de type 1 font ce qu'on appelle de **la paravirtualisation.**
Ils donnent aux systèmes invités l'accès au hardware sans émulation de celui-ci.
Les systèmes invités sont généralement modifiés pour que leur noyau puisse discuter avec le noyau de l'hyperviseur (noyaux dom0 et domU sous Xen Linux) et ainsi avoir accès directement au hardware.
XEN , Vmware ESX server , Hyper V (de microsoft) peuvent être rangés dans la catégorie d'hyperviseurs de type 1.

Les hyperviseurs de type 2 font ce qu'on appelle **de la virtualisation** (et oui sans le « para » devant), en fait à mon sens ce sont plus des **virtualiseurs** que des hyperviseurs.
Ce sont généralement des logiciels que l'on fait tourner sur un système d'exploitation existant.
Ce logiciel va nous permettre de créer des machines virtuelles.
Il émulera du matériel à la machine virtuelle comme par exemple une carte réseau , un contrôleur IDE/SCSI, une carte graphique etc ...
Et bien sûr tout ce travail d'émulation de périphériques à un coût supplémentaire en terme de ressources CPU et les performances de votre système virtuel vont s'en ressentir.
Les performances d'un système paravirtualisé seront beaucoup plus proches de celles d'un système réel que les performances d'un système virtualisé.
C'est pour cette même raison que l'on a vu apparaître ces derniers temps des processeurs avec des instructions spécifiques dédiées à la virtualisation (VT chez Intel et pacifia chez AMD).
C'est un peu comme si vous ajoutiez un deuxième processeur (mais ce n'est pas le cas!) à votre processeur central, qui aura en charge uniquement l'émulation de matériel pour machine virtuelle et qui libèrera votre processeur central de cette charge.
Les machines virtuelles y gagneront donc en performances avec ce type de processeur.
KVM (Kernel Virtual Machine), Vmware Server, Virtualbox, qemu, bochs peuvent être rangés dans la catégorie d'hyperviseur de type 2.

1.3. Quelques définitions

Définition d'un SAN : Le SAN repose sur la fibre optique, il est un réseau à part optimisé pour le transfert à haute vitesse de blocs de données vers et en provenance d'un disque. Les baies de disques peuvent être connectées au SAN, n'encombrant pas le réseau local (LAN) dédié aux utilisateurs.

Définition d'un HBA (Host Bus Adapter) : HBA (Host Bus Adapter) se présente généralement sous forme d'une carte PCI (PCI, PCI-X, PCI-e) et permet de connecter un serveur à un réseau de stockage (notamment un réseau SAN).

Définition d'un hyperviseur (wikipedia) : Un hyperviseur est un noyau hôte allégé et optimisé pour ne faire tourner que des noyaux d'OS invités, adaptés et optimisés pour tourner sur cette architecture spécifique. Les applications en espace utilisateur des OS invités tournent ainsi sur une pile de deux noyaux optimisés, les OS invités ayant conscience d'être virtualisés.
On dit aussi que 'est une couche d'abstraction qui s'installe entre le système d'exploitation et le matériel.

Logical Unit Number Dans ce document nous appellerons LUN tout disque du réseau de stockage attribué à un serveur.

Définition d'un volume physique : Les disques durs provenant d'un SAN et pris en charge par LVM (Logical Volume Management) sont appelés volumes physiques.

Définition d'un Volume Group : Un Volume Group est un ensemble de volumes physiques mis côte à côte.

Définition d'un logical Volume (ou volume logique) : Un volume logique est un espace découpé dans un Volume Group. Cet espace pourra alors être utilisé par le système pour y stocker un système de fichiers et des données. L'avantage du volume logique c'est qu'il peut être agrandi/réduit à chaud selon les limites de place du volume Group auquel il appartient.

Définition d'un snapshot : Faire un snapshot d'un volume logique, c'est faire une photo de celui-ci à un instant T. A l'instant T le snapshot ne contient que des pointeurs sur les données du volume logique original. A l'instant T+1 lorsque le contenu du volume logique original aura changé, le snapshot stockera alors les différences entre la photo prise à l'instant T et le contenu du volume original à T+1.

Système invité : Dans cet article on désignera généralement par "système invité" une machine virtuelle/paravirtuelle.

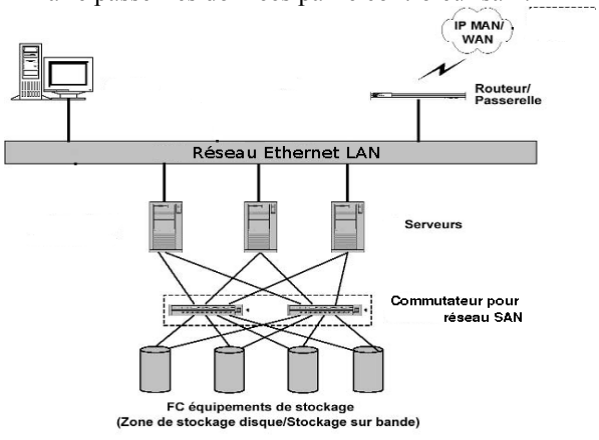
Système hôte : Dans cet article on désignera généralement par "système hôte" l'OS chargé de créer/gérer les systèmes.

2. Description du système d'information initial

Le système d'information du laboratoire dans lequel je travaille est constitué de nombreux serveurs sous Linux. Ces serveurs sont bien sûr reliés par un LAN Ethernet mais ils font également partie (par l'intermédiaire de deux adaptateurs HBA) d'un SAN (Stockage Area Network) sur lequel on trouvera une baie de disques et une librairie de sauvegarde.
Après délibérations, nous avons choisi d'utiliser les disques de la baie pour y stocker nos machines virtuelles.

Plusieurs avantages:

- Grâce à notre baie de disques configurée en RAID 5 les données sont sécurisées et un crash de disque sur la baie n'empêchera pas votre système virtuel de tourner.
- Les mêmes disques provenant d'un SAN peuvent être vus par **plusieurs serveurs/hyperviseurs** ce qui nous permettra de pouvoir faire de la **migration de machine virtuelle à chaud entre hyperviseurs**.
- **Le multipath :** comme la plupart de nos serveurs possèdent deux adaptateurs HBA SAN, un disque sur le SAN attribué à un serveur sera vu comme deux périphériques distincts. Le démon multipath sous linux (développé par C Varoqui) saura reconnaître en ces deux périphériques un seul et même disque (grâce à son identifiant SCSI) et nous permettra de faire de la répartition de charge au niveau des entrées sorties à partir des deux contrôleurs HBA. De plus si un des deux contrôleurs venait à tomber en panne, votre système ne sera pas corrompu, car le démon multipath permettra de faire passer les données par le contrôleur sain.



3. Attribution d'un disque provenant du réseau de stockage à un serveur sous LINUX

Notre baie de disques, grâce à un logiciel propriétaire, peut rendre visible ses disques par l'intermédiaire du SAN à un ou plusieurs serveurs linux. On dit souvent plus classiquement que l'on **attribue une LUN à un serveur physique.**
Lorsque vous attribuez une LUN à un serveur sous linux, la détection de celle-ci n'est pas automatique.
Sous le kernel 2.6 vous trouverez la plupart des informations connues du système sur votre adaptateur HBA dans ce répertoire : **/sys/class/fc_host/hostX** (où X est le numéro de votre adaptateur HBA)
Pour forcer votre système à scanner les périphériques qui lui sont attribués tapez la commande suivante :

```
echo 1 > /sys/class/fc_host/hostX/issue_lip
```

(donc quand vous disposez de deux adaptateurs HBA vous tapez la commande pour chacun d'eux) suivi d'un petit dmesg qui devrait vous indiquer si le disque en question a été vu par le système.
La commande **cat /proc/scsi/scsi** vous indiquera également quels sont les périphériques scsi/ san vus par votre système.

```
Attached devices :
Host: scsi0 Channel: 00 Id: 00 Lun: 00
Vendor: HITACHI Model: DF600F Rev: 0000
Type: Direct-Access ANSI SCSI revision: 03
Host: scsi0 Channel: 00 Id: 00 Lun: 01
Vendor: HITACHI Model: DF600F Rev: 0000
Type: Direct-Access ANSI SCSI revision: 03
Host: scsi1 Channel: 00 Id: 00 Lun: 00
Vendor: HITACHI Model: DF600F Rev: 0000
Type: Direct-Access ANSI SCSI revision: 03
Host: scsi1 Channel: 00 Id: 00 Lun: 01
Vendor: HITACHI Model: DF600F Rev: 0000
Type: Direct-Access ANSI SCSI revision: 03
```

Ici le système nous indique qu'il voit deux disques à partir de l'adaptateur scsi0 et deux disques à partir de l'adaptateur scsi1.Il s'agit en fait des deux mêmes disques vus par deux adaptateurs différents.

Sous debian il existe un utilitaire qui s'appelle scsiadd qui permet comme son nom l'indique d'ajouter des disques un par un.
scsiadd 1 0 0 1 va rajouter la ligne:

```
Host: scsi1 Channel: 00 Id: 00 Lun: 01
Vendor: HITACHI Model: DF600F Rev: 0000
```

```
Type: Direct-Access ANSI SCSI revision: 03
```

dans /proc/scsci/scsi

Une fois la LUN vue par votre système vous allez vouloir installer votre machine virtuelle sur celle-ci...

Deux choix alors s'offrent à vous:

Soit vous prenez la LUN telle qu'elle est et vous la partitionnez pour un système linux.

Soit vous transformez votre LUN en volume physique LVM,

```
pvcreate /dev/sdX
```

vous intégrez celui-ci à un volume group nommé vgvirtualdebian par exemple

```
vgcreate vgvirtualdebian /dev/sdX
```

Et enfin vous créez un volume logique d'environ 70 % de la capacité de votre LUN.

```
lvcreate -n lvsysteme -L <taille> vgvirtualdebian
```

Comme son nom l'indique ce sera le volume logique lvsysteme qui sera présenté en tant que disque système pour votre machine virtuelle.
L'avantage de cette solution est qu'elle permet de pouvoir faire des snapshots de votre système virtuel.

Une opération risquée à tester sur votre système linux virtuel ?
Un petit snapshot de votre système virtuel avec la commande :
Et on démarre une nouvelle instance de machine virtuelle à partir de lvsnap :-)

```
lvcreate --snapshot -L <taille> -name lvsnap
/dev/vgvirtualdebian/lvsysteme
```

Besoin d'une sauvegarde de votre machine virtuelle mais pour des raisons de disponibilité vous ne pouvez pas vous permettre d'arrêter les services qui y tournent ?
Idem, créez un snapshot de votre système virtuel et effectuez la sauvegarde à partir du snapshot

4. Notion de multichemins ou multipath sous linux

Comme je l'ai spécifié plus haut, les serveurs de notre infrastructure, pour la plupart sont pourvus de deux cartes fibre (HBA).
Une lun sera donc vue par nos serveurs linux comme deux périphériques distincts.

Si ce n'est pas évident pour vous, imaginez que vous achetez un disque pourvus de deux connecteurs IDE.
Chaque contrôleur IDE de votre carte mère sera relié au disque par l'intermédiaire d'une nappe.
Le fait que une de vos deux nappes IDE vienne à être défaillante ne sera pas gênant car vous pourrez toujours accéder à vos données par l'intermédiaire de la nappe IDE fonctionnelle (par contre pas de bol si les deux nappes ou les deux contrôleurs IDE viennent à être défaillants).
Or rappelez vous, tout périphérique sous UNIX/LINUX est défini par un fichier dans le répertoire /dev
Pour accéder alors au disque par l'intermédiaire de la première nappe le système va utiliser /dev/sda.
Et pour y accéder par l'intermédiaire de la deuxième nappe le système utilisera /dev/sdb.

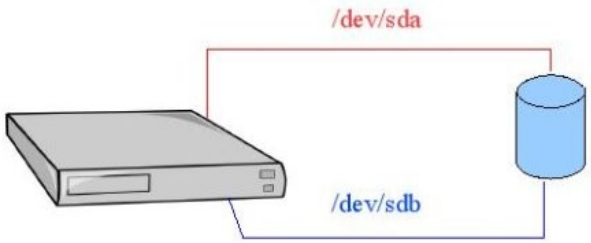


Illustration simplifiée du multipath

Si on pousse un peu plus loin la réflexion, on peut se dire qu'on pourra configurer notre système de façon à ce qu'il utilise /dev/sda par défaut donc de façon à ce qu'il utilise la première nappe IDE. En cas de défaillance de celle-ci le système commutera automatiquement sur la deuxième nappe.
C'est le principe du **failover.**

Maintenant imaginez que vous possédez deux disques durs pourvus chacun de deux connecteurs cette fois non plus IDE mais SCSI. Imaginez également que vous disposez sur votre carte mère de deux contrôleurs SCSI et que sur chacun de ces contrôleurs SCSI vous ayez branché une nappe capable de brancher deux périphériques (ça tombe bien, on possède deux disques durs). Vous pouvez donc connecter sur chaque contrôleur SCSI les deux disques durs.

Le premier contrôleur verra donc le premier disque dur comme /dev/sda et le deuxième disque dur comme /dev/sdb.
Le deuxième contrôleur verra le premier disque dur comme /dev/sdc et le deuxième comme /dev/sdd.

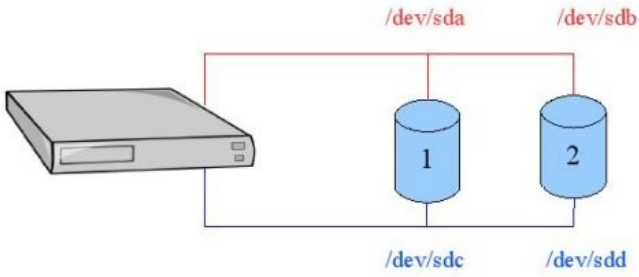


Illustration du multipath multi disques

Admettons qu'une importante opération d'écriture de données ait lieu sur le premier disque. L'envoi de données au disque se fait par l'intermédiaire de la première nappe, donc le système envoie des données sur sda.
Pendant ce laps de temps le système a un besoin d'écriture sur le deuxième disque. Comme la première nappe est déjà occupée par les données envoyées au premier disque le système prendra le chemin le moins encombré pour aller écrire sur le deuxième disque. Il s'agira donc de la nappe 2.
Dans ce cas là ce n'est plus du failover mais plutôt de la **répartition de charge.**

4.1. Le mapping de périphérique :

On l'a compris, le fait d'avoir deux contrôleurs fibre (HBA) sur nos serveurs nous permet de faire du multipathing (failover et/ou répartition de charge).
Mais concrètement comment ça va se passer au niveau de notre système linux ?
Prenons le cas précédent où le premier disque dur vu par le système (cf schéma ci-dessus) était accessible par /dev/sda et /dev/sdc.
Le démon multipath va faire appel au module dm-mod (device

mapper) pour créer un mapping de périphérique, c'est-à-dire une couche intermédiaire, entre le disque tel qu'il est vu par le système et les données écrites réellement sur le disque.
Lorsque le système voudra écrire des données sur le disque, il adressera le nom de périphérique mappé.

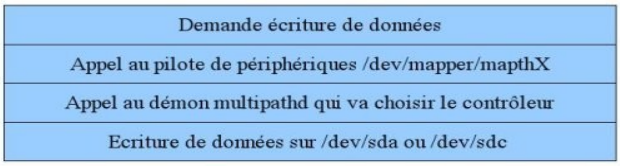


Illustration du fonctionnement du mapping de périphériques

Pour la lecture des données le principe sera le même.
En fait il suffit d'adresser les données à /dev/mapper/mpathX.
Pour connaître la valeur de X il suffit de taper la commande **/sbin/multipath -ll** sous root.

5. L'hyperviseur XEN

5.1. Virtualisation et paravirtualisation

Sous XEN il y a deux façons de faire
- Virtualisation totale.
- Paravirtualisation

Dans le premier cas le système invité n'a pas conscience d'être virtualisé puisque le système de virtualisation lui émule un vrai bios ainsi que du matériel comme une carte graphique, carte son, carte réseau , controleur IDE, etc...
Ainsi on peut prendre son cd et installer sa distribution linux préférée comme s'il s'agissait d'une machine réelle.

Dans le deuxième cas, le système a conscience d'être un système invité puisque celui-ci a été adapté pour être paravirtualisé.
Le noyau utilisé pour faire tourner notre système paravirtualisé a été spécialement conçu pour communiquer avec le noyau de l'hyperviseur.
Cette fois pas question de faire voir au système une carte graphique, son etc ... On ne s'embarrassera pas avec ce genre de détail. Un système linux peut très bien fonctionner sans carte graphique.
Tout ce dont on a besoin après tout c'est d'un système pouvant communiquer par le réseau et qui ait assez de ressources pour faire tourner nos applications.

L'avantage de la paravirtualisation est que ses performances sont très proches d'un système réel (plus proches que dans le cas d'une virtualisation totale).
Autre avantage : **la flexibilté.**

5.2. Limites de la virtualisation totale

Comme je le disais plus haut la virtualisation totale fait croire au système virtuel qu'il fonctionne sur une **vrai machine.** Au niveau stockage le système virtuel verra deux contrôleurs IDE (un primaire et un secondaire).
Malheureusement ces deux contrôleurs IDE nous limitent donc à quatre disques par machine virtuelle.
Ceci n'est malheureusement pas satisfaisant pour nos besoins de développement/production.
Chez nous, une machine qu'elle soit réelle ou virtuelle peut être amenée à faire tourner beaucoup d'applications au cours de sa vie.
Ces applications grandissent, requièrent plus d'espace disque et nous ne sommes pas capable par avance d'évaluer les besoins finaux en terme de stockage.
D'où l'intérêt de pouvoir attribuer de l'espace disque supplémentaire à la volée et sans limitation à une machine afin de pouvoir agrandir les espaces alloués aux applications à la demande.

Il se peut que à l'heure où j'écris ces lignes cette limitation soit maintenant levée.
Il me semble que sous KVM (Kernel Virtual Machine : la virtualisation native Linux) qui tout comme xen en mode virtualisation totale utilise une version modifiée de qemu, il est possible désormais d'ajouter des disques à chaud et que ceux-ci soient considérés comme disques SCSI par le système invité.
Par contre je ne sais pas ce qu'il en est pour xen.

5.3. Le choix entre la virtualisation totale et la paravirtualisation

En mode paravirtuel, il est possible :
- d'attribuer des disques supplémentaires à chaud au système invité,
- d'augmenter la mémoire RAM à chaud,
- d'attribuer de la cpu supplémentaire,
- d'attribuer une nouvelle interface réseau à la volée.

Bref la paravirtualisation représentait à nos yeux de meilleures performances et une flexibilité supplémentaire non négligeable.
Le choix n'aura donc pas été long, nos systèmes invités seront donc paravirtualisés.

5.4. Notion de domaine 0, domaine utilisateur (domU) :

Pour transformer votre système linux en hyperviseur, vous devez installer un noyau spécifique patché XEN.
La plupart des distributions fournissent ce noyau dans le système de packages.
Lorsque ensuite vous bootez votre machine sur le noyau XEN, celui-ci va démarrer le système installé sur les disques durs de votre serveur physique et le considérer comme le premier système invité.
Il s'agit du domaine 0, le système invité à partir duquel vous allez pouvoir contrôler tous les autres.
Les systèmes invités créés à partir du domaine 0 seront les domaines utilisateurs ou plus communément appelés domU.
Dans la branche Xen 2.X il fallait un noyau spécifique pour le domaine 0 et un noyau pour domU.
Aujourd'hui vous pouvez (mais ça n'est pas une obligation) utiliser le noyau du domaine 0 pour démarrer votre domU.
Quoiqu'il en soit, il faudra toujours que le noyau que vous utilisez pour démarrer votre domU soit stocké sur le dom0.

5.5. Notion de réseau Naté et réseau bridgé :

C'est une notion que l'on retrouve dans la plupart des virtualiseurs/ hyperviseurs que ce soit KVM, Vmware ou encore xen.

Ce qu'il faut retenir pour le mode NAT (Network Address Translation), **c'est que la machine/l'hyperviseur sur lequel tourne votre système invité fera office de passerelle.** Chaque fois que le système invité voudra envoyer des paquets sur le réseau, ceux-ci seront d'abord envoyés à la passerelle pour y subir une petite transformation.
Lorsque la passerelle recevra des paquets IP provenant du système invité et à destination du réseau elle va tout simplement remplacer l'adresse IP source de ce paquet (donc l'adresse du système invité) par la sienne (l'adresse IP de la passerelle) et ensuite envoyer le paquet sur le réseau.
Ainsi le destinataire du paquet aura toujours l'impression que le paquet a été émis directement de la passerelle et n'aura donc aucune connaissance de la machine virtuelle.

Retrouvez la suite de l'article de Cédric Tintanet en ligne : Lien85

Les derniers tutoriels et articles

Intégrer Ogre à Qt

Ce tutoriel est destiné à illustrer l'intégration du moteur 3D Ogre à Qt. Les concepts utilisés ici sont tout à fait utilisables pour d'autres moteurs (comme IrrLicht) et/ou pour d'autres toolkit graphique (comme WxWidget ou gtkmm).

1. Introduction

1.1. Pourquoi intégrer Ogre à Qt?

Interagir avec un jeu ou une scène 3D se fait couramment par le biais d'interfaces. L'utilisation majoritaire de ces interfaces est plutôt à chercher du côté de la conception d'interface in-game (tels les systèmes d'inventaires ou les menus que vous pouvez voir dans n'importe quel jeu). A ces interfaces, il faut ajouter ce que les concepteurs vont utiliser, c'est-à-dire des outils qui vont leur faciliter la création de scènes ou de niveaux, de matériaux, etc... Vous pouvez penser à l'éditeur Aurora de Neverwinter Nights, ou encore à l'Unreal Editor.
Ces deux besoins d'interfaces appellent deux réponses différentes. Pour le premier cas d'utilisation (interfaces in-game), on peut trouver des frameworks très performants, tel CEGUI (Lien86), qui remplissent parfaitement leur rôle de framework d'interfaces dédiées aux menus de jeux. Cependant, pour des besoins plus lourds tels les éditeurs, les fonctionnalités fournies ne répondent plus vraiment aux besoins des concepteurs, et plus particulièrement des concepteurs d'outils. C'est pourquoi, afin de gagner en concision, en souplesse et afin de ne pas réinventer constamment la roue, il est indispensable d'intégrer le moteur graphique dans un framework de conception d'applications (tels Qt, WxWidgets, gtkmm, .Net etc...).
Ce tutoriel s'appliquera, vous l'aurez deviné, à l'intégration du moteur 3D Ogre dans Qt. Plus que l'intégration, nous verrons par la suite quelques cas standards (bien que basiques) d'utilisation.

1.2. Pré-requis

- Une installation de Qt4 fonctionnelle
- une installation d'Ogre3D fonctionnelle (tuto réalisé avec la 1.4.5 et 1.4.8)
- En plus des bases en Qt (signaux/slots, création de widget), il est utile de connaître un minimum Ogre.

Avant d'entrer plus loin dans ce sujet, voici les tutoriaux qu'il est bon d'avoir suivi; les mécanismes présentés ici s'appuient sur des concepts basiques, mais les mécanismes spécifiques à Qt et Ogre seront assez peu abordés

- Tutoriaux Ogre sur DVP (Lien87)
- Tutoriel avancé n°2 (Lien88) et n°3 sur le Wiki Ogre (Lien89) (utilisation de RaySceneQuery utilisé dans la partie IV de ce tuto)

2. Initialiser le système

L'objectif de cette section sera d'obtenir un widget affichant effectivement ce que Ogre dessine. A la fin nous aurons donc un écran noir ainsi que la possibilité de changer la couleur de fond de notre viewport.
L'approche que je vais aborder ici est extrêmement simple, dans un souci de réduire le code montrant l'essentiel. Nous allons créer un widget stockant l'intégralité du système Ogre (un objet Root, scene manager etc...). Il est évident que dans un vrai code il est de loin préférable de découpler tout ceci, mais ce ne devrait pas être un problème une fois que vous aurez compris les principes. Il y a toutefois une précaution a prendre avant d'entrer dans le sujet. N'importez jamais le namespace Ogre dans le namespace global. Faire ceci vous mènera droit à des conflits de types définis en plusieurs endroits (par Ogre et Qt donc). Après vous avoir montré l'en-tête de la classe, je détaillerai les méthodes étape par étape.

En tête utilisé pour l'initialisation du système

```
class OgreWidget : public QWidget
{
    Q_OBJECT

public:
    OgreWidget(QWidget *parent = 0);
    ~OgreWidget();

public slots:
    void setBackgroundColour(QColor c);

protected:
    virtual void paintEvent(QPaintEvent *e);
    virtual void resizeEvent(QResizeEvent *e);
    virtual void showEvent(QShowEvent *e);

private:
    void initOgreSystem();

private:
    Ogre::Root         *ogreRoot;
    Ogre::SceneManager *ogreSceneManager;
    Ogre::RenderWindow *ogreRenderWindow;
    Ogre::Viewport     *ogreViewport;
    Ogre::Camera       *ogreCamera;
};
```

2.1. Objectif: écran noir (mais grâce à Ogre!)

Je vais détailler dans un premier temps la création du widget qui servira de réceptacle à l'affichage. Ce widget doit dériver de QWidget (Lien90) (surpris? :)), et réimplémenter quelques méthodes; le minimum étant *showEvent*, *paintEvent*, *moveEvent* et *resizeEvent* (toutes 4 sont des méthodes virtuelles protégées).
Ce widget devra avoir la propriété permettant de ne pas laisser le système dessiner un fond automatiquement: *Qt::WA_OpaquePaintEvent* et de ne pas laisser Qt gérer un double buffer: *Qt::WA_PaintOnScreen*.

```
OgreWidget::OgreWidget(QWidget *parent)
:QWidget(parent),
ogreRoot(0), ogreSceneManager(0), ogreRenderWindow(0),
ogreViewport(0),
ogreCamera(0)
{
    setAttribute(Qt::WA_OpaquePaintEvent);
    setAttribute(Qt::WA_PaintOnScreen);
```

```
    setMinimumSize(240,240);
}
```

Nous allons réimplémenter *showEvent* afin d'initialiser le système Ogre. En effet, son initialisation dépend du fait que nous ayons un handle système pour la fenêtre. Or cet handle n'est récupérable que lorsque la fenêtre est créée au niveau du système, et le moment le plus sûr est sa première apparition.

```
void OgreWidget::showEvent(QShowEvent *e)
{
    if(!ogreRoot)
    {
        initOgreSystem(); // l'initialisation d'ogre
est détaillée plus bas
    }

    QWidget::showEvent(e);
}
```

paintEvent sera réimplémenté afin de demander un rafraîchissement du viewport d'Ogre. En effet, si Ogre n'est pas maître de la boucle d'événement (ce qui va être le cas), il faut demander explicitement la mise à jour des viewports utilisés. De plus, on informe le reste du moteur qu'une nouvelle frame est dessinée.

```
void OgreWidget::paintEvent(QPaintEvent *e)
{
    ogreRoot->_fireFrameStarted();
        ogreRenderWindow->update();
    ogreRoot->_fireFrameEnded();

    e->accept();
}
```

Il est aussi nécessaire de réimplémenter *resizeEvent* et *moveEvent* afin de signifier à Ogre un changement au niveau de la résolution, et prendre en compte le nouvel aspect pour la caméra.

```
void OgreWidget::moveEvent(QMoveEvent *e)
{
    QWidget::moveEvent(e);

    if(e->isAccepted() && ogreRenderWindow)
    {
        ogreRenderWindow->windowMovedOrResized();
        update();
    }
}

void OgreWidget::resizeEvent(QResizeEvent *e)
{
    QWidget::resizeEvent(e);

    if(e->isAccepted())
    {
        const QSize &newSize = e->size();
        if(ogreRenderWindow)
        {
            ogreRenderWindow->resize(newSize.width(),
newSize.height());
            ogreRenderWindow->windowMovedOrResized();
        }
        if(ogreCamera)
        {
            Ogre::Real aspectRatio =
Ogre::Real(newSize.width()) /
Ogre::Real(newSize.height());
            ogreCamera->setAspectRatio(aspectRatio);
        }
    }
}
```

Passons maintenant à l'initialisation d'Ogre. Si vous avez déjà programmé avec Ogre sans utiliser le framework d'exemple, vous vous y retrouverez très certainement. Dans le cas contraire, il y aura quelques différences avec un développement se basant sur ledit framework. Nous allons nous même créer un objet *Ogre::Root*, ainsi que le *Ogre::RenderingSystem* et le *Ogre::SceneManager*. L'ensemble du code concernant l'initialisation d'Ogre est celui de la fonction **OgreWidget::initOgreSystem()**.
La création de *Ogre::Root* est simplissime. Nous utiliserons ici le constructeur par défaut. Il faudra donc que vous fournissiez les fichiers "ogre.cfg" (bien que le contenu de celui-ci ne soit pas utilisé et n'est donc pas indispensable physiquement) et "plugins.cfg" (celui-ci reste par contre indispensable).

```
void OgreWidget::initOgreSystem()
{
    ogreRoot = new Ogre::Root();
```

Le *RenderSystem* se crée en demandant à root une instance d'un rendering manager correspondant à un nom donné. Ici, j'utiliserai directement le nom du rendersystem opengl pour rester cross platform et que nous n'ayons pas à proposer une liste de rendering system. Lors de l'appel à Root::initialise (Lien91), notez que nous spécifions que nous ne voulons pas qu'Ogre crée une fenêtre automatiquement.
Si vous voulez le faire dans votre propre outil, il vous faudra en récupérer la liste grâce à Ogre::Root::getAvailableRenderers (Lien92).

```
    Ogre::RenderSystem *renderSystem = ogreRoot-
>getRenderSystemByName("OpenGL Rendering Subsystem");
    ogreRoot->setRenderSystem(renderSystem);
    ogreRoot->initialise(false);
```

Il n'y a pas de difficulté particulière au niveau de la création du scene manager. Ici encore, je code en dur le nom d'un SM standard afin de simplifier le code d'exemple. Pour proposer à l'utilisateur tout les SM disponibles, il vous faudra utiliser Ogre::SceneManagerEnumerator (Lien93). Contrairement aux rendersystems, il vous faudra obtenir un itérateur sur les méta-données des SceneManager, grâce à Ogre::SceneManagerEnumerator::getMetaDataIterator (Lien94). Une fois déréférencé, cet itérateur vous permettra d'obtenir un Ogre::SceneManagerMetaData (Lien95) qui contiendra tout ce qui permet de décrire un SceneManager.

```
    ogreSceneManager = ogreRoot-
>createSceneManager(Ogre::ST_GENERIC);
```

Vient maintenant le moment d'embarquer une vue Ogre au sein d'une fenêtre Qt : la création d'une *Ogre::RenderWindow*. Comme pour tout les objets vus précédemment, c'est sur demande à l'objet Root qu'une renderwindow est instanciée. Je vous encourage à jeter un oeil à la doc de cette méthode (Lien96). Nous allons donc devoir remplir une NameValuePairList dans laquelle nous spécifierons l'attribut *externalWindowHandle*, et bien sûr nous ne demanderons pas un affichage fullscreen.

```
    Ogre::NameValuePairList viewConfig;
    size_t widgetHandle;
#ifdef Q_WS_WIN
    widgetHandle = (size_t)((HWND)winId());
#else
    QWidget *q_parent = dynamic_cast <QWidget *>
(parent());
    QX11Info xInfo = x11Info();

    widgetHandle = Ogre::StringConverter::toString
((unsigned long)xInfo.display()) +
        ":" + Ogre::StringConverter::toString
((unsigned int)xInfo.screen()) +
```

```
        ":" + Ogre::StringConverter::toString
((unsigned long)q_parent->winId());
#endif
    viewConfig["externalWindowHandle"] =
Ogre::StringConverter::toString(widgetHandle);
    ogreRenderWindow = ogreRoot-
>createRenderWindow("Ogre rendering window",
                width(), height(), false, &viewConfig);
```

Il ne reste plus après qu'à créer une caméra (sur demande au SceneManager) et un viewport (sur demande auprès de l'objet RenderWindow nouvellement créé).

```
    ogreCamera = ogreSceneManager-
>createCamera("myCamera");

    ogreViewport = ogreRenderWindow-
>addViewport(ogreCamera);
    ogreViewport-
>setBackgroundColour(Ogre::ColourValue(0,0,0));
    ogreCamera->setAspectRatio(Ogre::Real(width()) /
Ogre::Real(height()));
}
```

2.2. Première interaction: modification de la couleur de fond

Les lecteurs attentifs auront remarqué dans l'en-tête la présence d'un slot **setBackgroundColour(QColor c)**. J'ai choisi QColor comme type afin de faciliter l'intégration avec le reste de Qt, et tout particulièrement QColorDialog (Lien97). Le seul problème de cette boîte de dialogue est qu'elle ne nous permettra pas de mettre à jour la vue en temps réel (elle est exécutée de façon modale, et n'émet pas de signaux correspondant à un changement de couleurs). Mais à titre d'exemple, cela suffira amplement. L'intérêt du slot est nul dans ce cas de figure, mais si vous faites votre propre boîte de dialogue non modale, simplement avoir un slot à connecter à un signal sera un plaisir!
Il y a une chose utile à connaître ici : QColor stocke et permet d'obtenir la couleur grâce à un seul appel sous le format AARRGGBB. Ogre nous permet de spécifier une couleur dans ce format, cela nous évite donc de récupérer et spécifier chaque canal.

```
void OgreWidget::setBackgroundColour(QColor c)
{
    if(ogreViewport)
    {
        Ogre::ColourValue ogreColour;
        ogreColour.setAsARGB(c.rgba());
        ogreViewport->setBackgroundColour(ogreColour);
    }
}
```

Pour profiter de tout ça, il ne vous reste plus qu'à construire une QMainWindow avec notre OgreWidget pour widget principal. Pour spécifier la couleur, vous pouvez créer une action dont l'activation affichera une QColorDialog, et vous permettra d'en récupérer la valeur pour la spécifier à l'instance de OgreWidget.
Une archive contenant tout le code nécessaire à cette partie est disponible (Lien98). Prenez simplement garde à configurer les fichiers qt_ogre.pro et plugins.cfg selon vos besoins, et n'oubliez pas de le mettre dans le répertoire d'exécution de votre application.

3. Affichage d'un objet et gestion de la caméra

Le but à la fin de cette section sera d'être capable de gérer la position et l'orientation de la caméra à la souris, au clavier ou par des spinbox placées dans un dock widget. La première partie est dédiée à avoir un objet qui s'affiche. Le gros du travail sera fait dans la partie 2 où j'aborderai chacune des méthodes d'entrées.

3.1. Ajouter de la lumière et un objet

Au code présenté ci-dessus, il va nous falloir ajouter une fonction qui va initialiser les ressources (pour l'objet et son material) ainsi qu'une fonction qui va créer la scène. La fonction initialisant les ressources n'est pas très intéressante en elle-même, il vous suffit de reprendre le code du framework d'exemple (ExempleApplication::setupResources dans le fichier ExampleApplication.h). Nous pourrions utiliser QSettings pour lire le fichier, mais il n'y a pas grand chose à y gagner ici. La scène créée sera très simple; il s'agit juste d'une lumière ambiante et d'un objet non animé.
Le chargement des ressources et la création de la scène se font à la fin de notre fonction d'initialisation. Ce sont des manipulations purement liées à Ogre et l'intégration dans un toolkit graphique n'a pas la moindre importance. Vous devriez obtenir l'image suivante en compilant et en exécutant l'archive fournie ici (Lien99). Dans cette dernière, j'ai ajouté le besoin du plugin Cg ainsi que les matériaux et objets nécessaires.

3.2. Gérer la caméra (gestion des événements et utilisation d'un dockwidget)

Cette partie va vous permettre de voir plusieurs façons de manipuler un objet. Ici, nous allons voir comment déplacer la caméra de trois façons différentes:

- Utilisation de spinbox
- Utilisation du clavier
- Utilisation de la souris

Une fois que nous aurons vu le positionnement, vous ne devriez pas avoir de problème pour implémenter l'orientation. La première modification à apporter concerne la classe OgreWidget. En effet, quelle que soit la méthode, nous allons devoir communiquer par le biais d'un signal la nouvelle position de la caméra. Le but ici est de conserver les spinbox synchronisées avec la position de la caméra, quelque soit la méthode qui sert au déplacement. Voici la signature dudit signal:

```
signals:
 void cameraPositionChanged(const Ogre::Vector3 &pos);
```

Retrouvez l'article de Denys Bulant en ligne : Lien100

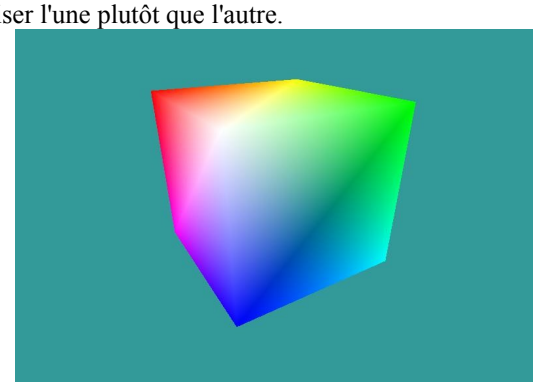
La géométrie en OpenGL : vers les VertexBufferObject

Pourquoi faire un tutoriel sur la géométrie en OpenGL ? Pourquoi faut-il s'y intéresser ? La réponse est simple, les technologies 3D sont sans cesse en cours d'évolution et d'améliorations. OpenGL ne fait pas exception, il y a régulièrement de nouvelles extensions pour améliorer ses performances et son efficacité. L'une d'elle se concentre sur l'amélioration de la géométrie : les Vertex Buffer Object. Comment en est on arrivé là ?

1. Introduction

Vous trouvez que votre application est lente ? Peut-être avez vous choisi un mauvais système pour rendre tous vos modèles 3D dans votre application ou votre jeu. Ce tutoriel a pour objectif de vous présenter toutes les techniques pour rendre des objets en OpenGL avec leurs avantages et leurs inconvénients. Nous commencerons par une présentation de la première méthode utilisée par OpenGL pour rendre des objets, le mode immédiat. Nous verrons en quoi ce système n'est pas forcément adapté au rendu et pourquoi les "VertexArray" ont fait leur apparition. Enfin, nous verrons en détail l'utilisation de la dernière, la plus récente et nouvelle technologie que sont les "Vertex Buffer Object".

Au travers de ce tutoriel, nous verrons le fonctionnement général de chacune des technologies au travers de l'exemple simple qu'est le dessin d'un cube coloré. Nous ferons une petite étude du coût de chaque méthode pour pouvoir les comparer et savoir dans quels cas utiliser l'une plutôt que l'autre.



Rendu de notre cube

2. Les prémices d'OpenGL : la géométrie immédiate

2.1. Rappel sur le dessin immédiat

La première méthode utilise, pour dessiner des formes 3D dans notre application, les fonctions "**immédiates**". Ce type de dessin est le premier que l'on apprend lorsque que l'on découvre l'API OpenGL. Cette méthode est dite "*immédiate*" car elle consiste en une succession d'appels de fonctions qui dessineront directement à l'écran. Pour décrire la géométrie, on utilise un ensemble de fonctions qui décrira toutes les composantes de notre objet 3D dessiné.

Chaque description de forme est encadrée par les fonctions ***glBegin()*** et ***glEnd()*** qui permettent de délimiter notre dessin. La première prend comme paramètre le type de forme à dessiner. Voici une description de ces fonctions :

Prototype de la fonction	Description
void ***glBegin***(GLenum mode);	Marque le début du dessin d'un groupe de primitives. *mode* précise le type des primitives parmi **GL_POINTS**, **GL_LINES**, **GL_LINE_STRIP**, **GL_LINE_LOOP**, **GL_TRIANGLES**, **GL_TRIANGLE_STRIP**, **GL_TRIANGLE_FAN**, **GL_QUADS**, **GL_QUAD_STRIP** et **GL_POLYGON**.
void ***glEnd***(void);	Marque la fin du dessin d'un groupe de primitives.

Chaque primitive est découpée en *vertex* qui correspondent à un sommet de notre forme 3D. Nous pouvons définir plusieurs types d'informations par vertex comme sa position, sa couleur, ... Chaque information est définie par un appel de fonction correspondant. Par exemple, nous utiliserons ***glVertex*()*** pour définir la position du vertex et ***glColor*()*** pour en définir la couleur. Ces fonctions sont décrites ici :

Prototype de la fonction	Description
void ***glVertex****(TYPE paramètres);	Spécifie la position du vertex à dessiner
void ***glNormal****(TYPE paramètres);	Spécifie la normale du vertex
void ***glColor****(TYPE paramètres);	Spécifie la couleur du vertex
void ***glSecondaryColor****(TYPE paramètres);	Spécifie la seconde couleur du vertex
void ***glIndex****(TYPE paramètres);	Spécifie l'index de la couleur du vertex
void ***glTexCoord****(TYPE paramètres);	Spécifie les coordonnées de la texture principale du vertex
void ***glMultiTexCoord****(TYPE paramètres);	Spécifie les coordonnées de texture en cas de "multi-texturing" du vertex
void ***glFogCoord****(TYPE paramètres);	Spécifie les coordonnées de fog du vertex

* doit être remplacé par la description du **TYPE** de données passé en paramètre.

2.2. Utilisation dans notre exemple

Pour l'exemple du cube, que nous allons suivre tout au long du tutoriel, nous allons créer un cube dont chacun des vertex a une couleur différente. Nous allons constituer notre cube de 6 faces. Chacune des faces est constituée de 2 triangles dont chaque vertex à une couleur différente. Nous obtenons donc un cube de 12 triangles.

Dans le cas que nous étudions actuellement, nous allons choisir le mode de primitives *GL_TRIANGLES* pour rendre notre objet. Ainsi, pour un cube, nous devons appeler 36 fois la fonction *glVertex*()* (12 triangles * 3 vertex). Il faut multiplier ce nombre par la quantité d'information par vertex pour obtenir le nombre total d'appels de fonction. Dans un cas extrême qui paramètrerait une couleur par vertex, la normale au vertex, des coordonnées de fog et les coordonnées pour deux textures, nous obtiendrions 180 appels de fonctions pour un simple cube avec 12 triangles. Pour notre exemple, nous nous contenterons d'une seule propriété de couleur par vertex donc nous obtiendrons 72 appels de fonctions plus les 2 appels pour *glBegin* et *glEnd*. Enfin, trois valeurs par appel de fonction nous donnent un total de 216 valeurs envoyées au processeur graphique.

Rendu d'un cube en mode immédiat

```
glBegin( GL_TRIANGLES );

        // gauche (x=-1)
        glColor3f( 1.0f, 0.0f, 0.0f );
glVertex3f( -1.0f, 1.0f, -1.0f );
        glColor3f( 1.0f, 0.0f, 1.0f );
glVertex3f( -1.0f, -1.0f, -1.0f );
        glColor3f( 1.0f, 1.0f, 1.0f );
glVertex3f( -1.0f, 1.0f, 1.0f );
        glColor3f( 1.0f, 1.0f, 1.0f );
```

```
glVertex3f( -1.0f, 1.0f, 1.0f );
        glColor3f( 1.0f, 0.0f, 1.0f );
glVertex3f( -1.0f, -1.0f, -1.0f );
        glColor3f( 0.0f, 0.0f, 1.0f );
glVertex3f( -1.0f, -1.0f, 1.0f );

        // droite (x=1)
        glColor3f( 0.0f, 1.0f, 0.0f );
glVertex3f( 1.0f, 1.0f, 1.0f );
        glColor3f( 0.0f, 1.0f, 1.0f );
glVertex3f( 1.0f, -1.0f, 1.0f );
        glColor3f( 1.0f, 1.0f, 0.0f );
glVertex3f( 1.0f, 1.0f, -1.0f );
        glColor3f( 1.0f, 1.0f, 0.0f );
glVertex3f( 1.0f, 1.0f, -1.0f );
        glColor3f( 0.0f, 1.0f, 1.0f );
glVertex3f( 1.0f, -1.0f, 1.0f );
        glColor3f( 1.0f, 1.0f, 1.0f );
glVertex3f( 1.0f, -1.0f, -1.0f );

        // bas (y=-1)
        glColor3f( 0.0f, 0.0f, 1.0f );
glVertex3f( -1.0f, -1.0f, 1.0f );
        glColor3f( 1.0f, 0.0f, 1.0f );
glVertex3f( -1.0f, -1.0f, -1.0f );
        glColor3f( 0.0f, 1.0f, 1.0f );
glVertex3f( 1.0f, -1.0f, 1.0f );
        glColor3f( 0.0f, 1.0f, 1.0f );
glVertex3f( 1.0f, -1.0f, 1.0f );
        glColor3f( 1.0f, 0.0f, 1.0f );
glVertex3f( -1.0f, -1.0f, -1.0f );
        glColor3f( 1.0f, 1.0f, 1.0f );
glVertex3f( 1.0f, -1.0f, -1.0f );

        // haut (y=1)
        glColor3f( 1.0f, 0.0f, 0.0f );
glVertex3f( -1.0f, 1.0f, -1.0f );
        glColor3f( 1.0f, 1.0f, 1.0f );
glVertex3f( -1.0f, 1.0f, 1.0f );
        glColor3f( 1.0f, 1.0f, 0.0f );
glVertex3f( 1.0f, 1.0f, -1.0f );
        glColor3f( 1.0f, 1.0f, 0.0f );
glVertex3f( 1.0f, 1.0f, -1.0f );
        glColor3f( 1.0f, 1.0f, 1.0f );
glVertex3f( -1.0f, 1.0f, 1.0f );
        glColor3f( 0.0f, 1.0f, 0.0f );
glVertex3f( 1.0f, 1.0f, 1.0f );

        // arriere (z=-1)
        glColor3f( 1.0f, 1.0f, 0.0f );
glVertex3f( 1.0f, 1.0f, -1.0f );
        glColor3f( 1.0f, 1.0f, 1.0f );
glVertex3f( 1.0f, -1.0f, -1.0f );
        glColor3f( 1.0f, 0.0f, 0.0f );
glVertex3f( -1.0f, 1.0f, -1.0f );
        glColor3f( 1.0f, 0.0f, 0.0f );
glVertex3f( -1.0f, 1.0f, -1.0f );
        glColor3f( 1.0f, 1.0f, 1.0f );
glVertex3f( 1.0f, -1.0f, -1.0f );
        glColor3f( 1.0f, 0.0f, 1.0f );
glVertex3f( -1.0f, -1.0f, -1.0f );

        // avant (z=1)
        glColor3f( 1.0f, 1.0f, 1.0f );
glVertex3f( -1.0f, 1.0f, 1.0f );
        glColor3f( 0.0f, 0.0f, 1.0f );
glVertex3f( -1.0f, -1.0f, 1.0f );
        glColor3f( 0.0f, 1.0f, 0.0f );
glVertex3f( 1.0f, 1.0f, 1.0f );
        glColor3f( 0.0f, 1.0f, 0.0f );
glVertex3f( 1.0f, 1.0f, 1.0f );
        glColor3f( 0.0f, 0.0f, 1.0f );
glVertex3f( -1.0f, -1.0f, 1.0f );
        glColor3f( 0.0f, 1.0f, 1.0f );
glVertex3f( 1.0f, -1.0f, 1.0f );
glEnd();
```

2.3. Avantages / Inconvénients de cette technique

Notre premier exemple permet de mettre en évidence certains avantages et certains inconvénients de la technique de dessin "*Immédiat*". Nous allons donc commencer par évoquer les avantages d'une telle technique :

- Permet de définir indépendamment chacune des informations des vertex
- Permet d'augmenter la lisibilité du code
- Regroupe clairement chaque rendu par type de primitives

Malgré ceux-ci, on remarque rapidement un grand nombre d'inconvénients très gênants pour des applications "temps-réel". En voici une liste non-exhaustive :

- Un grand nombre d'appels de fonction pour des géométries simples
- Grande répétitivité de commandes identiques
- Complexifie les structures de données utilisables
- Obligation de séparer chacune des composantes de nos informations (X, Y et Z pour les positions de nos vertex, par exemple)

La technique de dessin "*Immédiat*" est très contraignante du point de vue performance pour une application "temps-réel". Les deux problèmes principaux sont que le transfert des données est extrêmement lent au vu de la quantité d'appel de fonctions et du peu d'informations envoyées au GPU par chaque appel. Le second problème vient du nombre d'appels identiques à chaque frame. OpenGL nous propose une nouvelle technique permettant de résoudre notamment le nombre d'appels de fonctions, de limiter l'envoi d'information en double. Cette technique utilise le principe des "*vertex array*". De plus, cette technique pourra être complétée par la méthode des "Display List" qui évitera un grand nombre d'appels de fonction répétitifs à chaque frame dans le cas de géométrie "statique".

Retrouvez la suite de cet article en ligne (Lien104)

3. Conclusion

Nous avons donc vu les différentes optimisations qu'a apportées OpenGL au rendu de géométrie. Chacune des techniques a ses avantages propres mais les **"Vertex Array"** et les **"Vertex Buffer Object"** ont pour objectif de combler les lacunes des précédentes. A titre d'exemple, sur une carte Nvidia GeForce 7600 GS, les dessins de notre cube nous donnent les performances suivantes :

Immédiat	Vertex Array	Vertex Buffer Object
1445 FPS	1485 FPS	1515 FPS

Cet exemple n'est pas vraiment significatif car il se contente de dessiner 12 triangles mais l'on remarque déjà une légère différence en faveurs de VBO. Pour augmenter les performances de votre application OpenGL, vous pouvez trouver quelques informations dans notre FAQ :
FAQ Optimisation OpenGL (Lien101)

Une des dernières extensions d'OpenGL (GL_EXT_draw_instanced (Lien102)) permet le dessin de type "instancing". Cette extension permet de dessiner plusieurs fois le même objet avec un seul appel de fonction.

Pour toutes questions, n'hésitez pas à demander directement sur le forum OpenGL (Lien103)

Retrouvez la suite de l'article de Cyril Doillon en ligne : Lien104

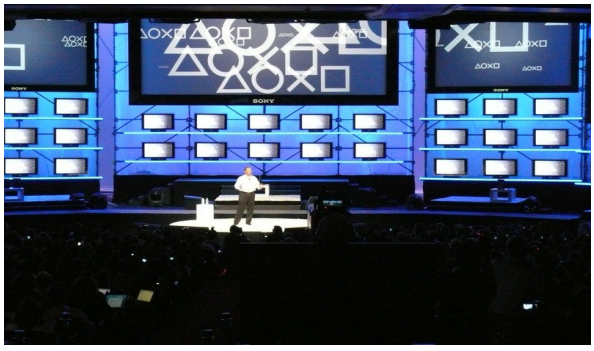
Retour sur l'E3

Comme annoncé il y a quelques jours, l'E3 s'est déroulé au Convention Center de Los Angeles. Ayant perdu de son succès face à la GameConvention de Leipzig et au Tokyo Game Show, le salon essaye de reprendre des couleurs. Cette année peut peut-être le montrer grâce à quelques annonces intéressantes. Comme vous le savez surement, il y a eu trois grandes conférences des trois leaders du marchés : **Microsoft, Nintendo et Sony** qui ont présenté chacun leurs nouveautés.

Lors de la conférence Microsoft, de grands noms du jeu-vidéo sont apparu comme **Fallout3, Resident Evil 5, Gears of War 2 et Fable 2** sur Xbox360. Ces titres annonces une continuité dans la stratégie à succès de Microsoft. Ce succès peut prouver par des chiffres comme, par exemple, **un nouvel abonné au Xbox Live toutes les 5 secondes**. Le Xbox Live va également subir quelques modifications notamment par l'agrandissement de son catalogue de vidéo à la demande. L'interface globale va également être remaniée. Enfin, il a été annoncé que le nouveau **Final Fantasy XIII** sera présent sur la plateforme de Microsoft, ainsi, Sony perd une grande exclusivité.

La conférence Nintendo a également été une source, modeste, de nouveautés. Il a donc été présenté deux nouveaux périphériques que sont **le Wii MotionPlus et le Wii Speak** qui respectivement permettront d'améliorer la détection de mouvement de la Wiimote et la communication entre différents joueurs à distance. Le successeur de Wii Sport a été présenté sous le nom de **Wii Sports Resort** qui permettra de toucher des sports comme le Jet-Ski, le Freesbee et le Kendo. Annoncé depuis le début de la Wii, **Wii Music** a enfin montré le bout de son nez. Enfin, une grande nouvelle est l'annonce d'un Grand Theft Auto sur la console portable Nintendo DS. Seul son nom a été communiqué à savoir : **Grand Theft Auto, Chinatown Wars**.

La conférence de Sony s'est vu rassurante pour les actionnaires : des chiffres, beaucoup de chiffres sur ses différentes plateformes : Playstation 2, Playstation 3 et Playstation Portable... Quelques jeux prévus, la sortie d'une PS3 avec 80go qui remplacera celle actuel en Europe pour 399$. On sortira notamment des noms comme **Resistance 2** et le très attendu **LittleBigPlanet** ainsi que le prochain **Ratchet & Clank Future : Quest for Booty**. Un MMO PC et PS3 a été présenté sous le nom de **DC Universe Online**.



Voila, l'E3 est fini, pensons au prochain grand évènement : la **Game Convention** de Leipzig

Retrouvez ce billet sur le blog Jeux : Lien105

Les derniers tutoriels et articles

Application d'un algorithme génétique au problème du voyageur de commerce

Le but de cet article est d'illustrer l'implémentation d'un algorithme génétique sur un exemple concret de recherche opérationnelle : le problème NP complet dit du voyageur de commerce. Prérequis : algorithmes génétiques, programmation objet, C++

1. Introduction

Rappel : Le problème du voyageur de commerce est le suivant :

- soient N villes séparées chacune d'elles par une distance D(i,j)
- trouver le chemin le plus court passant par toutes les villes et retournant à son point de départ.

En d'autres termes, il s'agit de déterminer le plus court chemin hamiltonien sur un graphe donné. La taille de l'espace de solutions est 1/2(N-1)!.

L'utilisation d'un algorithme génétique est particulièrement adaptée à cet exemple vu l'indépendance des solutions potentielles, la simplicité de créer des solutions potentielles et vu sa formalisation 'relativement' simple. Dans cet article, je vais proposer une implémentation en C++, la plus générique possible, adaptable à d'autres problèmes.

2. Déroulement global de l'algorithme

2.1. Evolution des générations

Une approche objet nous fournit une abstraction du déroulement de l'algorithme pour ne plus lier les spécificités du problème qu'au type des individus considérés. De manière tout à fait transparente, le déroulement est le suivant :

- création d'une population aléatoire
- pour chaque génération
- déterminer une liste d'individus à muter
- faire muter ces individus
- déterminer une liste d'individus à croiser
- croiser ces individus
- injecter ces 2 nouvelles listes d'individus dans la population
- choisir les individus pour la génération suivante

L'élément principal, auquel bénéficiera l'abstraction du type des individus, possède la logique macroscopique de l'algorithme. Il est à la fois central et très simple : il ne possède comme seul membre que la population en cours de traitement. Cependant, libre à vous de l'enrichir si le besoin s'en fait sentir.

```
    void runNgenerations(int nbGenerations){
        int size = int(population.size());

        int itmp = 0;
        for (int i = 0; i<nbGenerations; i++){
            // sélection des meilleurs éléments
            std::set<individual> toMutate;
            selectIndividuals( int(size /1.5), toMutate);

            // croisements / mutations
            std::set<individual> mutated;
            mutate(toMutate, .4, mutated);

            std::set<individual> toCross;
            selectIndividuals( size /2, toCross);

            std::set<individual> crossed;
            cross(toCross, .8, crossed);

            // insertion dans population
            injectIntoPopulation(mutated);
            injectIntoPopulation(crossed);

            troncatePopulation(size);

            individual &tmp = *population.begin();
            std::cout << i << " " << tmp.fitness << " " << tmp.display() << std::endl;
        }
    }
```

Le conteneur d'individus sera très souvent sollicité pour ajouter, rechercher et supprimer de nouveaux éléments. L'utilisation d'un conteneur trié s'avèrera des plus utiles. L'unicité des individus n'est pas nécessaire, cependant des doublons risquent d'augmenter le volume de données et le temps de calcul, bien qu'ils puissent aussi favoriser certains individus lors des choix aléatoires. J'ai choisi d'utiliser un std::set<individual>

Cet élément central, qui sera le moteur de l'algorithme génétique possède toutes les fonctions générales de manipulation de la population : création, sélection, insertion, suppression.

```
void injectIntoPopulation(std::set<individual>& l){
        for (std::set<individual>::iterator i = l.begin(); i != l.end(); i++)
                population.insert(*i);

    }

    void troncatePopulation(int n){
        int nbFound = 0;
        std::set<individual>::iterator i = population.begin();
        while (i != population.end() && nbFound < n){
            i++;
            nbFound++;
        }

        if (i != population.end())
            population.erase(i, population.end());
    }
```

2.2. Eviter la régression à tout prix

Le but étant quand même d'améliorer les solutions proposées, nous devons nous prémunir d'éventuelles régressions. J'ai choisi de me baser sur le meilleur élément d'une génération pour juger de

l'avancement de la résolution. Bien qu'il puisse s'agir d'un extremum local sans chance d'amélioration, cet individu correspond à la meilleure solution trouvée au problème et en est la réponse si l'exécution devait s'arrêter.

La méthode basique consiste à toujours reprendre les meilleurs individus d'une génération sur l'autre, on est sûr que la génération suivante sera au moins aussi bonne que la précédente. On évite le risque de perdre les meilleurs individus par une sélection malchanceuse les oubliant, malgré leur meilleure qualité. Oui, mais qu'en est-il des autres individus moins bons ? Ils sont peut-être la clef de la convergence optimale sans que nous le sachions encore. C'est pourquoi la solution de sécurité consiste à ne jeter aucun individu d'une génération sur l'autre. Evidemment je vous laisse imaginer le temps de calcul qui va croître de génération en génération.

La solution naïve serait de faire une simple sélection (roulette?) pour déterminer les individus de la prochaine génération, mais on amène le risque de régression. Cette solution a l'avantage de rester vraiment neutre pour ne mettre en avant que les individus qui le méritent, sans a priori sur la solution finale. Un compromis pourrait être

Nouvelle génération = sélection[roulette?] (ancienne génération + éléments mutés + éléments croisés) + échantillon des meilleurs individus de la génération courante

Dans tous les cas nous sommes obligés de jeter certains individus, autant prendre le moins de risques.

La méthode de sélection est cependant libre. Je propose un système de roulette pour privilégier les meilleurs individus. Les meilleurs fitness étant décroissantes, j'ai juste ajouté un artifice pour inverser cette tendance.

```
    void selectIndividuals(int nbToSelect,
std::set<individual>& selected){
        int globalSum = 0;

        std::set<individual>::iterator it =
population.end(); it--;
        int maxFitness = it->fitness;

        for (std::set<individual>::iterator i =
population.begin(); i!=population.end(); i++)
            globalSum += maxFitness  - i-
>fitness;    // inversion du sens des fitness

        std::set<individual>::iterator ii =
population.begin();
        // conservation des 5 meilleurs individus
        for (int i=0; i<5; i++, ii++)
            selected.insert(*ii);

                // roulettes de sélection
        for (int i = 0; i<nbToSelect ; i++){
            int found = rand()%globalSum;
            int a=0;
            std::set<individual>::iterator ite =
population.begin();
            a += maxFitness  - ite->fitness;

            while (a<found){
                ite++;
                a += maxFitness  - ite->fitness;
            }

            if (selected.find(*ite) == selected.end())
                selected.insert(*ite);
        }
    }
```

On peut aussi simplement conserver les meilleurs éléments pour la sélection. Le conteneur est trié, il suffit de prendre les premiers éléments :

```
        std::set<individual>::iterator ii =
population.begin();
        for (int i=0; i<nbToSelect; i++){
            ii++;
            selected.insert(*ii);
        }
```

2.3. Création des individus

La création d'une population dépend fortement du problème, c'est encore un élément à basculer dans l'individu :

```
void individual::createRandomly(){
    int nbValues = int(data.distances.size());

    // on commence à la position 0
    solution.push_back(0);
    std::vector<int> remainingPossibilities;
    for (int i=1; i<nbValues; i++)
        remainingPossibilities.push_back(i);

    int a=1;

        // on recherche les villes suivantes
    while (a < nbValues){
        solution.push_back(data.selectNearCity(a));
        a++;
    }

        // on rend la solution cohérente avec le
problème
    makeCoherent();

    calculateQuality();
}
```

3. Définition du problème

Pour rester le plus simple possible, j'ai choisi de représenter le problème par le graphe des villes : une matrice aléatoire, symétrique et de diagonale nulle. Ainsi les distances sont équivalentes selon leur orientation. Vous pouvez bien sûr supprimer la symétrie, l'algorithme se débrouillera avec la matrice donnée.

Chaque problème vient avec sa définition de règles, ses données et ses contraintes. Le moteur d'algorithmes n'a pas besoin directement de connaître le problème à résoudre, seuls les individus doivent pouvoir vérifier les contraintes.

Chaque individu doit simplement avoir une référence ou un pointeur sur le jeu de données à résoudre. Dans notre exemple du voyageur de commerce, chaque individu possède une simple référence sur l'instance des règles.

Le problème peut simplement se décrire :

```
class problem{
public:
    problem(int);
    int selectNearCity(int);

    std::vector<std::vector<int> > distances;

protected:
    std::vector< std::map<int, int> > dual;
```

```
    std::vector< int > maxDistances;
    std::vector<int > globalSums;
};
```

On y trouve bien sûr les distances entre les villes (distances) mais aussi d'autres structures telles que les données duales ou encore les éléments de base des roulettes de sélection.

On retrouve aussi une petite fonction permettant de choisir une ville à proximité d'une ville donnée en tenant compte des données duales, implémenté sous la forme d'une roulette.

```
int problem::selectNearCity(int from){
    int f = rand()%globalSums[from];
    int i = 0;
    std::map<int, int>::iterator it =
dual[from].begin();
    i += maxDistances[from] - it->first;

    // roulette de sélection
        while (i<f && it!=dual[from].end()){
        i += maxDistances[from] - it->first;
        it++;
    }

        if (it == dual[from].end())
                return dual[from]
[ int(dual[from].size()-1) ];

    return it->second;
}
```

4. Spécialisation suivant le type d'individus

Un individu correspond à une solution possible au problème. Quel que soit le problème traité, on devra être en mesure d'attribuer une fitness ou une qualité à chacun de nos individus.
Le problème traité déterminera la logique microscopique : la création aléatoire, les méthodes de calcul de qualité, de mutations et de croisements des individus. Il suffit de les énoncer pour voir apparaître des fonctions virtuelles pour ces opérations.

- createRandomly
- calculateQuality
- mutate
- cross

Voilà, le squelette est posé, reste maintenant à spécifier les traitements relatifs à notre problème du voyageur de commerce.

Dans le cas du voyageur de commerce, chaque individu est composé d'une liste de villes. Sa qualité est donnée par la distance totale du chemin qu'il représente. Ce problème est mono critères, la fitness de chaque individu reflète directement sa résolution du problème, ce qui simplifie la résolution et l'implémentation.

```
class individual{
public:
    individual(problem &);    // constructeur à partir
du jeu de données
    individual(const individual&);
    individual& operator =(const individual&);
    ~individual();

    void createRandomly();    // création aléatoire
    void mutate();                // mutation
    individual operator*(individual&);    // croisement

    std::vector<int> solution;        // chemin retenu
dans le graphe
    int fitness;                            // qualité
de la solution
    problem &data;        // référence sur les données
du problème

    std::string display();
    void makeCoherent();    // vérifier que this répond
bien au problème

protected:
    void calculateQuality();
};
```

La mutation d'un individu peut être toute opération aléatoire qui le fait passer indépendamment à un autre chemin hamiltonien. On peut par exemple permuter un ou plusieurs couples de villes pour retomber sur un chemin hamiltonien potentiellement meilleur. On peut considérer la mutation de deux manières : elle peut créer un nouvel individu qui correspondra à l'individu muté ou bien elle pourra directement modifier l'individu muté. La première approche permet de conserver l'individu originel au cas où sa mutation l'aurait rendu moins bon. C'est ce que j'ai choisi dans mon implémentation.

```
void individual::mutate(){
    // intervertir 2 villes
    int l = int(solution.size());

        // on détermine le nombre de couples à
intervertir
    int nb = rand()%5;
    for (int i=0; i<nb; i++){
        int a = rand()%l;
        int b = rand()%l;

        int tmp = solution[a];
        solution[a] = solution[b];
        solution[b] = tmp;
    }

        // on détermine la fitness de la solution créée
    calculateQuality();
}
```

Le croisement de solutions peut être toute opération permettant d'obtenir une nouvelle solution à partir de 2 ou plus individus existants. J'ai choisi d'effectuer un croisement multipoints en reprenant des parties de chaque individu parent pour créer un nouvel individu. Il faut cependant toujours conserver la cohérence des solutions créées en rétablissant le caractère hamiltonien d'un individu créé. De manière plus évidente que pour la mutation, le croisement crée un nouvel élément. On aurait cependant pu modifier l'un des parents, amenant le même risque que dans la mutation sans conservation d'historique.

```
individual individual::operator*(individual& i){

    int l = int(i.solution.size());

    // croisement multipoints
    int a = rand()%l;
    int b = rand()%l;
    int min1, min2;

    min1 = (a<b)?a:b;
    min2 = (a<b)?b:a;

        // pour la première inversion de composantes
    for (int ii=0; ii<min1; ii++){
        int tmp =  solution[ii];
        solution[ii] = i.solution[ii];
```

```
        i.solution[ii] = tmp;
    }

        // pour la seconde inversion de composantes
    for (int ii=min1; ii<min2; ii++){
        int tmp =  i.solution[ii];
        i.solution[ii] = solution[ii];
        solution[ii] = tmp;
    }

        // pour la troisième inversion de composantes
(2 points de croisement = 3 composantes)
    for (int ii=min2; ii<l; ii++){
        int tmp =  solution[ii];
        solution[ii] = i.solution[ii];
        i.solution[ii] = tmp;
    }

        // on rend les individus cohérents
    makeCoherent();
    i.makeCoherent();

    calculateQuality();
    i.calculateQuality();

    return *this;
}
```

Pour profiter du tri de la population manipulée par le moteur d'algorithme génétique, on pensera à spécifier un opérateur de tri pour la classe d'individus manipulée :

```
bool operator<(individu& i1, individu& i2){
    return i1.fitness< i2.fitness;
}
```

Note : on remarque que les traitements aléatoires sont pléthore dans un algorithme génétique, n'hésitez pas à utiliser un autre générateur que rand() si le besoin s'en fait sentir.

5. De l'art de ne pas laisser la machine chercher naïvement

Si déjà on a réussi à reléguer les données du problème dans la classe individual pour abstraire le moteur d'algorithmes, autant faire que les individus utilisent effectivement ces données, et pas que pour calculer leur propre fitness.

Chacun des traitements des individus pourrait être orienté pour résoudre plus rapidement les contraintes du problème. Par exemple, la création aléatoire des individus pourrait privilégier des éléments consécutifs proches, de même les mutations pourraient essayer de relier des villes proches. La considération de la matrice des villes fait ainsi apparaître sa matrice duale, comportant pour chaque ville, celles qui sont les plus proches. Il suffit pour ça de chercher le successeur d'une ville par une roulette de sélection sur la matrice duale, en tenant toujours compte des villes déjà utilisées.

Ce processus de sélection par roulette peut être rattaché directement aux données du problème, par l'introduction d'un objet 'problème' à part entière, contenant toutes les données et les contraintes de notre graphe, y compris la fameuse matrice duale. Il suffit ensuite que les individus s'y réfèrent lorsqu'ils ont besoin de connaître une ville qui peut potentiellement suivre une ville donnée.

En agissant ainsi, on va s'économiser de nombreuses mutations ou croisements perdus d'avance, ou des individus visiblement trop mauvais. Mais évidemment, le revers de la médaille concernera le comportement de l'algorithme. En élaguant trop largement des solutions mauvaises au premier abord, on prendra le risque de passer à côté de la solution optimale en s'enfermant dans un extremum local.

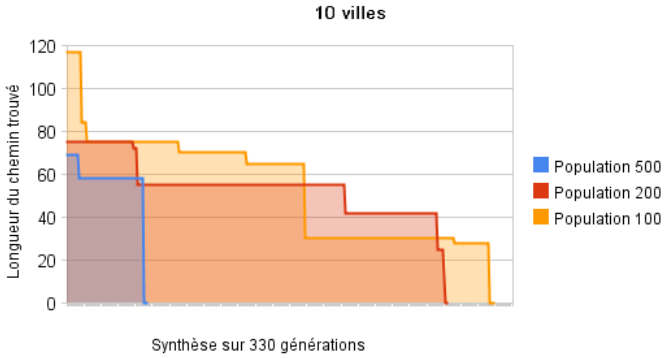
6. Diagrammes de convergence

Le programmeur a la main sur plusieurs paramètres pour affiner la recherche de solutions :

- La taille de la population
- L'évolution de la taille de la population

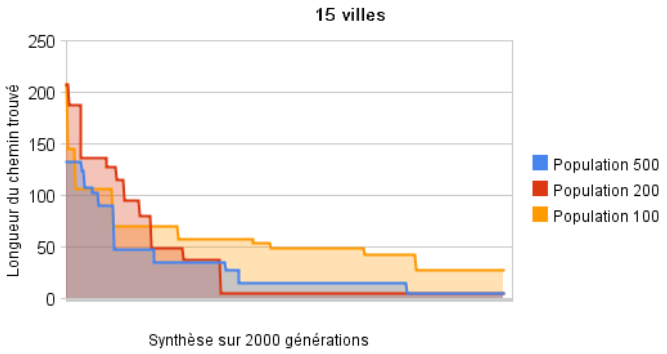
Le hasard étant omniprésent, les exécutions ne démarrent pas toutes avec le même meilleur individu, le départ des courbes n'est donc pas le même partout. J'ai choisi de manipuler des populations de taille fixe, voici les résultats que j'ai pu obtenir.

6.1. Recherche sur 10 villes



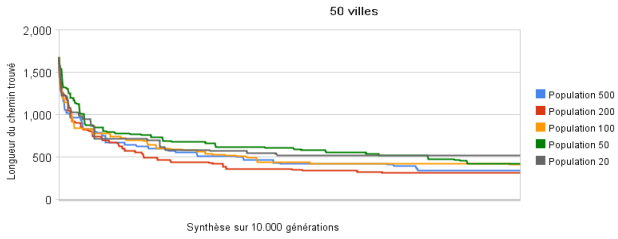
Différentes exécutions avec des populations de 100, 200 et 500 ont toutes convergé assez rapidement. On remarquera cependant que la convergence avec 500 individus a nécessité moins de générations. Autre point notable : l'exécution avec une population de 100 a été meilleure que l'exécution avec 200 individus pendant un court instant, le hasard est facétieux.

7.2. Recherche sur 15 villes



Pour un espace de recherche 200.000 fois plus grand que précédemment, les exécutions n'ont pas convergé en 2000 générations, cependant les solutions proposées sont très très proches de la solution optimale (longueur 20 contre une longueur optimale de 15). Comme précédemment, la convergence est plus lente avec une population moins importante, même si on remarque que les exécutions avec 500 et 200 individus sont tour à tour meilleures.

7.3. Recherche sur 50 villes



Pour un espace de recherche près de 7.10^51 fois plus grand que précédemment, les exécutions n'ont pas convergé au bout de 10.000 générations. L'algorithme a réussi à diviser par 5 la taille du chemin hamiltonien depuis les individus aléatoires. J'ai stoppé l'exécution à 10.000 générations, mais la courbe de progression montre bien une amélioration continue des solutions.

Sur les diagrammes de convergence, les vitesses de progression sont rapides au début de l'exécution puis de plus en plus lente, ce qui peut justement nous inciter à arrêter prématurément l'exécution pour garantir un rapport qualité/temps optimal. Et sur cet exemple encore plus que sur les précédents, on se rend compte du caractère improbable des tailles des populations : les 5 exécutions fournissent des résultats similaires. D'un point de vue statistique, il aurait fallu d'un seul individu pour résoudre le problème, en une seule génération, mais on s'aperçoit quand même de l'avantage d'utiliser des populations nombreuses.

8. Conclusion

Bien que nécessitant beaucoup de temps à l'exécution, les algorithmes génétiques n'ont pas à avoir honte de leur prestation dans la résolution du problème du voyageur de commerce. N'étant pas faits pour fournir des solutions optimales, ils ont néanmoins réussi à en déterminer certaines. Bien que non optimales, les solutions proposées pour la résolution à 50 villes n'ont cessé de s'améliorer et s'amélioreraient encore si on poursuivait l'exécution.

Télécharger les sources C++ : implementation.zip (Lien106)

Retrouvez l'article de Pierre Schwartz en ligne : Lien107

Un TAD pour éditeurs de diagrammes

Cet article présente les *quadtrees*, une technique spécifique à la *segmentation en régions*, et se propose de les adapter au domaine toujours plus populaire de l'édition de diagrammes.

1. Introduction

Autrefois réservé à l'édition de schémas électroniques et au routage de circuits imprimés, l'éditeur de diagrammes est un utilitaire qui aujourd'hui ne dépareille plus dans la boîte à outils de l'informaticien moderne. Il est bien loin le temps où l'on se gaussait des ordigrammes qui, on en était certains, ne remplaceraient jamais le pseudo-code.

Car si les diagrammes ont bien échoué dans la représentation de bas niveau, leur succès n'en est pas moins grand dans la communication et la représentation de haut niveau.
Les domaines embrassés par cet engouement sont tellement multiples :

-
- diagrammes de classes (Lien108)
- diagrammes d'objets (Lien109)
- diagrammes syntaxiques
- réseaux sémantiques
- automates
- réseaux de PETRI
- théorie des graphes (Lien110)
- théorie des catégories et *diagram chasing*

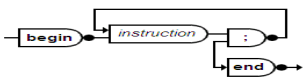
que l'ensemble du secteur est touché, du plus praticien au plus théoricien.

Considérant l'ampleur de cet engouement il paraît surprenant de constater le manque de maturité des outils disponibles.

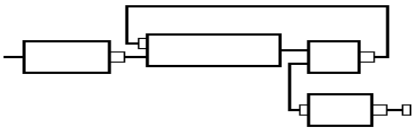
De simples listes chaînées ont toujours très bien fait l'affaire pour manipuler les fameuses fenêtres qui ont supplanté la ligne de commande. C'est pourquoi il a été tentant de croire qu'une structure linéaire était assez bien adaptée pour la représentation bi-dimensionelle. Le résultat ce sont des manipulations saccadées, des connexions flottantes, et plus généralement l'absence ou la pauvreté de l'assistance à la cohérence d'ensemble. Bref tout ce qui fait aujourd'hui la frustation à utiliser un éditeur de diagrammes. On a fini par accepter de peiner pour faire des schémas propres.

2. Domaine

D'une façon générale on peut dire que l'édition de schémas consiste à relier des boîtes par des connecteurs :



En effet, dans sa représentation interne, le schéma ci-dessus n'est pas fondamentalement différent du schéma ci-dessous :



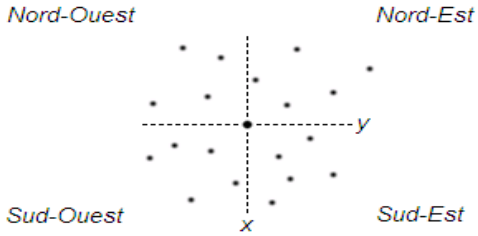
Où toutes les boîtes sont des rectangles et tous les connecteurs sont des lignes verticales (rectangles de largeur nulle) ou horizontales (rectangles de hauteur nulle).

3. Type Abstrait de Données

Il s'agit bien entendu d'une partition du plan dont le pivot serait un rectangle.

Cette définition nous rapproche du *quadtree*, un type abstrait de données (Lien111) qui partitionne le plan et dont le pivot est un point.

Un *quadtree* découpe le plan en 4 quadrants, *Nord-Ouest*, *Nord-Est*, *Sud-Ouest* et *Sud-est*.
Cette dichotomie du plan est parfaite car tout point est inclus dans l'un des 4 quadrants.



L'algorithme de *quadtree* est traditionnellement une méthode de traitement de l'image dont elle permet une segmentation en régions (voir le tutoriel Segmentation en régions (Lien112) par Xavier Philippeau).
Le logiciel Fractint (Lien113) (rendu d'images fractales) a popularisé le *quadtree* en l'utilisant pour accélérer le rendu d'images fractales. La fractale est récursivement découpée en 4 quadrants afin de détecter au mieux l'intérieur des régions connexes de l'image. Ces régions sont alors affichées par un simple remplissage de couleur plutôt que par un coûteux calcul point par point.

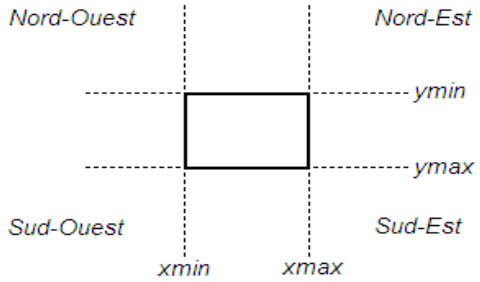
Alors pouvons-nous réutiliser la conception du *quadtree* pour

l'adapter au rectangle ?
Oui et non.

- oui, car chacun des 4 coins du rectangle défini naturellement son quadrant associé
- non, car la dichotomie ne pourra pas être parfaite, certains rectangles ne seront entièrement inclus dans aucun de ces 4 quadrants

Le rectangle ouvre dans le plan un 5ième espace de recherche en forme d'une croix centrée sur lui et qui s'étend aux 4 points cardinaux.
Un rectangle n'est inclus dans aucun quadrant si et seulement si son intersection avec cette croix n'est pas vide.
On est là en présence de deux définitions complémentaires de l'appartenance :

- l'appartenance à un *quadrant* est plus restrictive, elle se fait par *inclusion*
- l'appartenance à la *croix inter-quadrants* est plus large, elle se fait par simple *intersection*



Un rectangle est un intervalle à deux dimensions borné par *(xmin,xmax,ymin,ymax)*.
La dichotomie est possible car il existe une relation d'ordre partiel sur l'ensemble des intervalles.
Quant à la croix inter-quadrants, c'est le prix à payer pour l'absence d'une relation d'ordre total.

Une droite, ou un segment de droite, partitionne le plan en deux demi-plans.
Il n'y a donc pas d'obstacle théorique à inclure des segments quelconques (non parallèles aux axes) dans une structure de partition du plan. Nous avons toutefois écarté cette possibilité afin de ne pas encombrer le discours avec trop de cas particuliers.

Voyons maintenant comment définir efficacement les opérations habituelles sur les rectangles hiérarchisés par notre nouveau *TAD* (*Type Abstrait de Données*).

4. Recherche d'un rectangle

Une opération indispensable pour toute application interactive c'est la recherche.
Soit un point *(x,y)* du plan (typiquement le lieu d'un évènement de pointage), on veut retrouver efficacement le rectangle (c'est-à-dire la boîte) désigné par l'utilisateur (pour le sélectionner, le déplacer, le redimensionner...).

C'est typiquement le genre de requête qu'une structure dichotomique permet d'accélérer considérablement.
En effet, considérons un rectangle pivot :

- les chances sont grandes pour que le lieu du pointage se situe dans l'un des 4 quadrants, l'espace de recherche est alors divisé par 4
- dans le cas contraire le point est dans la croix et l'espace de recherche est alors amputé des 4 quadrants

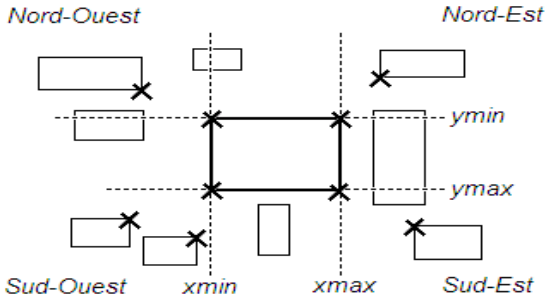
```
boucle
   choisir
   ou bien l'arbre est vide alors
      le point ne désigne aucun rectangle
   ou bien le point est dans le rectangle pivot alors
      on a trouvé le rectangle désigné
   ou bien le point est dans le quadrant NO alors
      on poursuit la recherche dans le sous-arbre NO
   ou bien le point est dans le quadrant NE alors
      on poursuit la recherche dans le sous-arbre NE
   ou bien le point est dans le quadrant SO alors
      on poursuit la recherche dans le sous-arbre SO
   ou bien le point est dans le quadrant SE alors
      on poursuit la recherche dans le sous-arbre SE
   sinon // le point est dans la croix
      on poursuit la recherche dans le sous-arbre NSWE
   fin choisir
fin boucle
```

5. Intersection avec un rectangle

Une autre opération indispensable pour une application interactive c'est l'intersection.
On veut ajouter une boîte, ou bien déplacer une boîte, et alors :

- soit l'application interdit que deux boîtes se superposent
- soit l'application adopte un rendu spécial lorsqu'une boîte est couverte par une autre

Dans les deux cas il faut d'abord identifier les deux rectangles qui se croisent.
Pour ce faire on suit la décomposition du plan selon nos 4 quadrants, pour décider si notre rectangle est dans un cadran il nous suffira de comparer le coin du rectangle pivot avec le coin qui lui est opposé par notre rectangle.




```
boucle
   choisir
   ou bien l'arbre est vide alors
      l'intersection est vide
   ou bien le rectangle croise le rectangle pivot alors
      on a trouvé une intersection
   ou bien le coin SE est dans le quadrant NO du pivot
alors
      on poursuit la recherche dans le sous-arbre NO
   ou bien le coin SO est dans le quadrant NE du pivot
alors
      on poursuit la recherche dans le sous-arbre NE
   ou bien le coin NE est dans le quadrant SO du pivot
alors
      on poursuit la recherche dans le sous-arbre SO
   ou bien le coin NO est dans le quadrant SE du pivot
alors
      on poursuit la recherche dans le sous-arbre SE
   sinon // le rectangle et la croix ont une
intersection non-vide
      on poursuit la recherche dans le sous-arbre NSWE
   fin choisir
fin boucle
```

6. Insertion d'un rectangle

Dernière opération indispensable: on veut ajouter une boîte dans notre arbre.

Même méthode que pour l'intersection: pour chaque coin du rectangle le coin opposé du rectangle pivot sert de discriminant.

```
boucle
   choisir
   ou bien l'arbre est vide alors
      son père est le père du rectangle à ajouter
   ou bien le rectangle croise le rectangle pivot alors
      on a trouvé une intersection
   ou bien le coin SE est dans le quadrant NO du pivot
alors
      on ajoute le rectangle au sous-arbre NO
   ou bien le coin SO est dans le quadrant NE du pivot
alors
      on ajoute le rectangle au sous-arbre NE
   ou bien le coin NE est dans le quadrant SO du pivot
alors
      on ajoute le rectangle au sous-arbre SO
   ou bien le coin NO est dans le quadrant SE du pivot
alors
      on ajoute le rectangle au sous-arbre SE
   sinon // le rectangle et la croix ont une
intersection non-vide
      on ajoute le rectangle au sous-arbre NSWE
   fin choisir
fin boucle
```

7. Conclusion

On pourra faire remarquer que parmi les 5 sous-arbres du *TAD* proposé :

- seuls 4 effectuent une opération de division comparable à un arbre de recherche
- le 5ième effectue une soustraction comparable à une liste chaînée

Il y a cependant une différence fondamentale entre notre sous-arbre *NSWE* et la queue d'une simple liste chaînée :

- à chaque noeud une liste chaînée ne soustrait qu'un simple rectangle à l'espace de recherche
- à chaque noeud notre espace en croix soustrait 4 quadrants à l'espace de recherche

Intuitivement notre *TAD* paraît donc une solution simple et efficace pour partitionner le plan à l'aide d'un ensemble de rectangles.

Annexe A. Implémentation avec Objective-Caml

Il fait chaud et la tentation est grande de programmer comme un chameau.
Le lecteur inaverti des dangers du désert pourra s'épargner une expérience pénible en choisissant de se contenter du pseudo-code plutôt que de s'aventurer dans ces contrées inhospitalières.

A. 1. Type Inductif persistent

Sans surprise, le type inductif est semblable à un arbre binaire de recherche.

Chaque noeud porte :

- les bornes d'un intervalle à deux dimensions
- les 4 sous-arbres pour l'inclusion dans les 4 quadrants *nw*, *ne*, *sw* et *se*
- le sous-arbre *nswe* pour l'intersection avec la croix centrée sur le rectangle pivot

```
type shape =
  | Nil
  | Rect of rect
and rect =
  {
  xmin:int;
  xmax:int;
  ymin:int;
  ymax:int;
  nw:  shape;
  ne:  shape;
  sw:  shape;
  se:  shape;
  nswe: shape;
  }
```

A.2. Notion de chemin dirigé

Ce qui nous importe vraiment c'est une façon de parcourir la structure de donnée.
Mais pas n'importe quelle façon, une façon particulière.
Celle qui partant de la racine de l'arbre suit le chemin désigné par le rectangle qui nous intéresse.
Nous appellerons *r* ce rectangle qui dirige le cheminement à travers un arbre *s*.

La fonction *directed_path* capture assez bien la notion de chemin dirigé par un rectangle *r* à travers un arbre *s* :

- la fonction *nil* dit que faire de l'arbre vide
- les fonctions *nw*, *ne*, *sw*, *se* et *nswe* disent que faire pour chacune des 5 directions possibles
- la fonction *overlap* dit que faire en cas de recouvrement de 2 rectangles

```
let directed_path nil nw ne sw se nswe overlap s r =
  let rec helper s =
    match s with
    | Nil -> nil r
    | Rect t ->
        if r.ymax < t.ymin then
          if r.xmax < t.xmin then nw (helper t.nw) t
          else if r.xmin > t.xmax then ne (helper t.ne)
t
          else nswe (helper t.nswe) t
        else if r.ymin > t.ymax then
          if r.xmax < t.xmin then sw (helper t.sw) t
          else if r.xmin > t.xmax then se (helper t.se)
t
          else nswe (helper t.nswe) t
        else overlap t
  in helper s
```

A.3. Prélude

Chaque chamelier a probablement un petit fichier *prelude.ml* contenant quelques définitions triviales.
En voici quatres qui figurent dans le mien, rien de bien méchant ici :

- *id1* est la fonction identité sur le 1ier argument
- *id2* est la fonction identité sur le 2nd argument
- *none* est un constructeur pour *rien*
- *some* est un constructeur pour *quelque chose*

```
let id1 x y = x
let id2 x y = y
let none x = None
let some x = Some x
```

Retrouvez la suite de l'article de Damien Guichard en ligne : Lien114

Liens

Lien1 : http://en.wikipedia.org/wiki/Higher-order_function
Lien2 : http://en.wikipedia.org/wiki/Currying
Lien3 : http://blog.developpez.com/djo-mos?title=introduction_a_scala_methodes_et_fonctio
Lien4 : http://xmlbeans.apache.org/
Lien5 : http://longbeach.developpez.com/tutoriels/XML/XMLBeans/
Lien6 : http://download.eclipse.org/technology/babel/update-site/
Lien7 : http://www.eclipse.org/cdt/
Lien8 : http://www.eclipse.org/cobol/
Lien9 : http://www.eclipse.org/dltk/
Lien10 : http://www.phpeclipse.de/tiki-view_articles.php
Lien11 : http://pydev.sourceforge.net/
Lien12 : http://texlipse.sourceforge.net/
Lien13 : http://e-p-i-c.sourceforge.net/
Lien14 : http://www.improve-technologies.com/alpha/esharp/
Lien15 : http://emonic.sourceforge.net/
Lien16 : http://rubyeclipse.sourceforge.net/
Lien17 : http://eclipseme.org/
Lien18 : http://eclipse.developpez.com/faq/
Lien19 : http://dico.developpez.com/html/1713-Generalites-POO-Programmation-Orientee-Objet.php
Lien20 : http://dico.developpez.com/html/1574-Conception-design-pattern.php
Lien21 : http://dico.developpez.com/html/219-Internet-HTTP-HyperText-Transmission-Protocol.php
Lien22 : http://dico.developpez.com/html/2322-Conception-flux-de-donnees.php
Lien23 : http://dico.developpez.com/html/270-Internet-service-web.php
Lien28 : http://www.zend.com/en/store/education/certification/yellow-pages.php
Lien29 : http://www.certificationphp.com/tests/evaluation
Lien30 : http://www.anaska.com/formations/formation-php-expert-certifie.php
Lien31 : http://www.amazon.fr/Architects-Zend-Certification-Study-Guide/dp/0973862149/ref=sr_1_2?ie=UTF8&s=english-books&qid=1212161752&sr=1-2
Lien32 : http://www.amazon.fr/Zend-Certification-Practice-Test-Book/dp/0973589884/ref=sr_1_3?ie=UTF8&s=english-books&qid=1212161752&sr=1-3
Lien33 : http://www.amazon.fr/Zend-PHP-Certification-Study-Guide/dp/0672327090/ref=sr_1_5?ie=UTF8&s=english-books&qid=1212161752&sr=1-5
Lien34 : http://www.phparch.com/c/product/vulcan/view
Lien35 : http://julien-pauli.developpez.com/tutoriels/php/zend-certif/
Lien36 : http://dico.developpez.com/html/832-Securite-certificat.php
Lien37 : http://dico.developpez.com/html/1453-Langages-API-Application-Programming-Interface.php
Lien38 : http://www.developpez.net/forums/member.php?u=116373
Lien39 : http://jcrozier.developpez.com/articles/web/panier/
Lien40 : http://dico.developpez.com/html/1589-Securite-SSL-Secure-Sockets-Layer.php
Lien41 : http://dico.developpez.com/html/1591-Internet-HTTPS-HTTPS-Secured.php
Lien42 : http://dico.developpez.com/html/832-Securite-certificat.php
Lien43 : http://dico.developpez.com/html/3046-Generalites-template.php
Lien44 : http://thierry-godin.developpez.com/php/atos/
Lien45 : http://www.w3.org/TR/REC-CSS2/selector.html
Lien46 : http://meyerweb.com/eric/tools/css/reset/
Lien47 : http://cssglobe.developpez.com/tutoriels/css/selection-css/
Lien48 : http://pckult.developpez.com/tutoriels/css/menu-digg/
Lien49 : http://gorgonite.developpez.com/tutoriels/fonctionnel/ocaml/introduction/
Lien50 : http://www.mono-project.com/
Lien51 : http://research.microsoft.com/fsharp/release.aspx
Lien52 : http://laurent.le-brun.eu/site/index.php/2008/05/24/36-how-to-use-fsharp-1-9-4-17-on-mono
Lien53 : http://www.microsoft.com/downloads/details.aspx?FamilyId=40646580-97FA-4698-B65F-620D4B4B1ED7&displaylang=en
Lien54 : http://sourceforge.net/projects/fsharp-mode
Lien55 : http://llb.developpez.com/articles/fsharp/presentation/
Lien56 : http://llb.developpez.com/articles/fsharp/types-de-base-et-expressions/
Lien57 : http://broux.developpez.com/articles/csharp/introduction-silverlight-2/
Lien58 : http://nico-pyright.developpez.com/tutoriel/vs2008/csharp/silverlightandmysql/gettodolist.php
Lien59 : http://www.cameronalbert.com/post/2008/03/HttpWebRequest-Helper-for-Silverlight-2.aspx
Lien60 : http://nico-pyright.developpez.com/tutoriel/vs2008/csharp/silverlightandmysql/
Lien61 : http://melem.developpez.com/reseaux/rawsockets/
Lien62 : http://vicenzo.developpez.com/ocilib/index.php?page=download
Lien63 : http://vicenzo.developpez.com/ocilib/download/ChangeLog-2.5.0.txt
Lien64 : http://vicenzo.developpez.com/ocilib/
Lien65 : http://orclib.sourceforge.net/
Lien66 : http://orclib.sourceforge.net/blog/
Lien67 : http://blog.developpez.com/oracle?title=nouvelle_version_v2_5_0_de_la_bibliotheq
Lien68 : http://arkham46.developpez.com/articles/office/clgdiplus/doc/
Lien69 : http://www.microsoft.com/downloads/details.aspx?FamilyID=6a63ab9c-df12-4d41-933c-be590feaa05a&DisplayLang=en
Lien70 : ftp://ftp-developpez.com/arkham46/articles/office/clgdiplus/doc/fichiers/clgdiplus.zip
Lien71 : http://arkham46.developpez.com/articles/office/clgdiplus/
Lien72 : http://cafeine.developpez.com/access/tutoriel/photos/
Lien73 : http://access.developpez.com/sources/?page=commondlg#ShowColor
Lien74 : http://access.developpez.com/faq/?page=Ctrl#survol_ctrl
Lien75 : http://access.developpez.com/faq/?page=zdl#AbsDsListe
Lien76 : http://argyronet.developpez.com/office/access/selectitemlistAB/
Lien77 : http://access.developpez.com/faq/?page=Divers#infobullevba
Lien78 : http://starec.developpez.com/tuto/barrecommande/
Lien79 : http://jeannot45.developpez.com/articles/access/gestionplanning/
Lien80 : http://www.cygwin.com/
Lien81 : http://www.straightrunning.com/XmingNotes/

Lien82 : http://www.chiark.greenend.org.uk/~sgtatham/putty/
Lien83 : http://come-david.developpez.com/tutoriels/remoteX/
Lien84 : http://fr.wikipedia.org/wiki/SCSI
Lien85 : http://cedric-tintanet.developpez.com/tutoriels/linux/xen-san/
Lien86 : http://www.cegui.org.uk/wiki/index.php/Main_Page
Lien87 : http://bauland.developpez.com/tutoriel/ogre/
Lien88 : http://www.ogre3d.org/wiki/index.php/Intermediate_Tutorial_2
Lien89 : http://www.ogre3d.org/wiki/index.php/Intermediate_Tutorial_3
Lien90 : http://doc.trolltech.com/4.3/qwidget.html
Lien91 : http://www.ogre3d.org/docs/api/html/classOgre_1_1Root.html#Ogre_1_1Roota10
Lien92 : http://www.ogre3d.org/docs/api/html/classOgre_1_1Root.html#Ogre_1_1Roota6
Lien93 : http://www.ogre3d.org/docs/api/html/classOgre_1_1SceneManagerEnumerator.html
Lien94 : http://www.ogre3d.org/docs/api/html/classOgre_1_1SceneManagerEnumerator.html#Ogre_1_1SceneManagerEnumeratora5
Lien95 : http://www.ogre3d.org/docs/api/html/structOgre_1_1SceneManagerMetaData.html
Lien96 : http://www.ogre3d.org/docs/api/html/classOgre_1_1Root.html#Ogre_1_1Roota34
Lien97 : http://doc.trolltech.com/4.3/qcolordialog.html
Lien98 : ftp://ftp-developpez.com/irmatden/tutoriels/fichiers/qt_ogre.zip
Lien99 : ftp://ftp-developpez.com/irmatden/tutoriels/fichiers/qt_ogre_objet_lumiere.zip
Lien100 : http://irmatden.developpez.com/tutoriels/qt/integration-ogre-qt/
Lien101 : http://jeux.developpez.com/faq/opengl/?page=optimisations
Lien102 : http://www.opengl.org/registry/specs/EXT/draw_instanced.txt
Lien103 : http://www.developpez.net/forums/f53/technologies-divers/developpement-2d-3d-jeux/apis-multimedia/opengl/
Lien104 : http://raptor.developpez.com/tutorial/opengl/vbo/
Lien105 : http://blog.developpez.com/jeux?title=bilan_de_l_e3
Lien106 : http://khayyam.developpez.com/articles/algo/voyageur-de-commerce/genetique/fichiers/implementation.zip
Lien107 : http://khayyam.developpez.com/articles/algo/voyageur-de-commerce/genetique/
Lien108 : http://dico.developpez.com/html/985-Conception-diagramme-de-classes.php
Lien109 : http://dico.developpez.com/html/984-Conception-diagramme-dobjets.php
Lien110 : http://dico.developpez.com/html/1774-Langages-graphe.php
Lien111 : http://dico.developpez.com/html/1748-Langages-TAD-type-abstrait-de-donnees.php
Lien112 : http://xphilipp.developpez.com/articles/segmentation/regions/?page=page_3#LIII
Lien113 : http://fr.wikipedia.org/wiki/Fractint
Lien114 : http://damien-guichard.developpez.com/tutoriels/algo/editeur-diagrammes-boites/