Developpez Magazine

Edition de Juin-Juillet 2007.
Numéro 10.
Magazine en ligne gratuit.
Diffusion de copies conformes à l'original autorisée.
Réalisation : Baptiste Wicht
Rédaction : la rédaction de Developpez
Contact : magazine@redaction-developpez.com

Index

Editorial

Le 10ème magazine de Developpez.com est arrivé. Retrouvez les nouveaux tutoriaux et ressources des rubriques de Developpez.com. Merci de votre fidelité !

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

La rédaction

Article Dotnet

Utilisez l'effet Glass de Vista

Windows Vista et ses fenêtres transparentes vous fait rêver ? Apprenez à utiliser le thème Aero Glass de Windows Vista pour que vous puissiez, vous aussi, avoir de la transparence dans vos fenêtres Winform.

par **Florian Casabiance**
Page 17

Article Mac

Decouvrez QuartzComposer

Certains développeurs excellent dans l'art de réaliser des effets graphiques impressionnants. Apple a pensé à tous les autres en livrant QuartzComposer avec Tiger, la dernière version de MacOS X.

par **Romain Guy**
Page 56

L'API java.nio du JDK 1.4

De nombreux programmeurs critiquent Java pour des raisons de performances. Malgré les améliorations constantes de nos ordinateurs et des JVM, ce reproche perdure. Nous allons cependant voir que connaître les API suffit largement pour obtenir d'excellentes performances.

1. Introduction

Aujourd'hui, les solutions pour accélérer un programme écrit en Java ne sont pas nombreuses. Nous pouvons changer de processeur pour un plus performant, utiliser l'une des dernières machines virtuelles ou compiler le code source en mode natif. Mais tout ceci se révèle bien souvent superflu. Prenons l'exemple des deux codes source populate1.java et populate2.java. Ceux-ci créent une chaîne de caractères contenant 10 000 entiers tirés au hasard. La première implémentation utilise un objet String et demande 39 secondes d'exécution sur un Pentium IV 1.6 Ghz nanti de 256 Mo de DDRAM. La seconde quant à elle, sur la même configuration, ne requière que 0.03 secondes. Le secret réside dans l'emploi de la classe StringBuffer, plus appropriée.

2. Opérations de lecture

L'exemple présenté ci-dessus s'avère particulièrement probant. Sachez qu'il en va de même pour de nombreux types d'opérations et notamment les opérations de lecture de fichiers. Depuis les premières versions du JDK, les classes de support des entrées/sorties résidaient dans le paquetage java.io. Avec l'apparition du JDK 1.4 est né le paquetage java.nio, acronyme signifiant "New Input/Output". Le rôle principal de ce paquetage consiste à améliorer les performances du vieillissant java.io ainsi que d'apporter de nouvelles fonctionnalités telles que le verrouillage des fichiers.

Les performances peuvent augmenter considérablement si l'on emploie cette nouvelle API. Néanmoins, nul n'est besoin de l'utiliser constamment. Pour déterminer si une application nécessite une optimisation de ses opérations de lecture ou d'écriture, nous pouvons faire appel à une option de la machine virtuelle qui réalise une trace de notre programme. Le fichier source SourceCodeLinesCounter0.java contient un programme parcourant récursivement l'intégralité des fichiers portant l'extension .java présents dans le répertoire courant et affichant le nombre de lignes total. L'exécution de notre outil sur les sources du JDK 1.4 nécessite 111 secondes. Que pouvons-nous faire pour améliorer ceci ? Exécutons tout d'abord l'option de trace :

```
java -Xrunhprof:cpu=sample,depth=15,file=prof0.txt
SourceCodeLinesCounter0
```

Cette commande génère un fichier nommé prof0.txt dans lequel nous pouvons lire les lignes suivantes :

```
1 87.76% 87.76%     724    19
java.io.FileInputStream.read
2  5.45% 93.21%      45    20
java.io.FileInputStream.open
```

Ceci signifie que 87% du temps d'exécution est alloué à l'exécution de la méthode read() de la classe FileInputStream. Nous allons tâcher d'optimiser les performances à cet endroit. En parcourant le contenu de java.io, nous découvrons la présence de la classe BufferedReader censée accélérer les opérations de lecture. Le code source SourceCodeLinesCounter1.java emploie cette dernière. Le temps d'exécution tombe alors à 13 secondes. Pouvons-nous faire mieux ? Avec les API antérieures au JDK 1.4, non, à moins d'invoquer la méthode readLine() plutôt que de faire un parcours caractère par caractère.

3. Les nouvelles classes

La nouvelle API met à disposition quatre nouvelles familles de classes : les Buffer, séquences linéaires de données, les Charset, faisant la transition entre caractères Unicode et séquences d'octets, les Channels, qui représente des tuyaux de communication bi-directionnels et enfin les Selector, employés pour des opérations d'E/S asynchrones sur des threads. Cette API introduit notamment la notion d'opérations de lecture et d'écriture non bloquantes. Nous ne nous préoccuperons ce mois-ci que des Buffer et Channel.

La classe MappedByteBuffer sert à créer une représentation complète d'un fichier en mémoire. De ce fait, son emploi autorise au système à procéder à des optimisations importantes et les temps d'accès s'en trouvent amoindris. Le fichier source SourceCodeLinesCounter2 fait appel à cette classe. Voici un exemple de son utilisation :

```
in = new FileInputStream("fichier").getChannel();
int size = (int) in.size();
bytes = in.map(FileChannel.MapMode.READ_ONLY, 0,
size);
char c = (char) bytes.get();
```

Les Buffer de l'API NIO nécessitent de créer un nouveau Channel que nous obtenons grâce à une instance de FileInputStream. La méthode map() des canaux autorise le mappage en mémoire d'une portion définie d'un fichier, ici la totalité. Cette implémentation diminue encore le temps d'exécution de notre programme pour le porter à 10 secondes.

Les Buffer possèdent des propriétés très intéressantes principalement en ce qui concerne le positionnement du pointeur de lecture/écriture. Nous pouvons aisément définir ou récupérer sa position courante. Un Buffer possède une capacité maximale et une limite de remplissage. Nous pouvons par exemple écrire 10 octets dans un buffer d'une capacité de 512 octets. Sa limite sera alors 10 octets. Trois méthodes se révèlent utiles : clear(), flip() et rewind(). La dernière replace le pointeur de lecture en début de séquence. La seconde place la limite sur la position courante et renvoie le pointeur en début de séquence. Enfin,

clear() renvoie le pointeur au début et place la limite en fin de séquence. Ces notions de position se veulent extrêmement importantes puisque chaque action de lecture (par l'entremise des méthodes get()) ou d'écriture (par l'intermédiaire des méthodes put()) fait avance le pointeur.

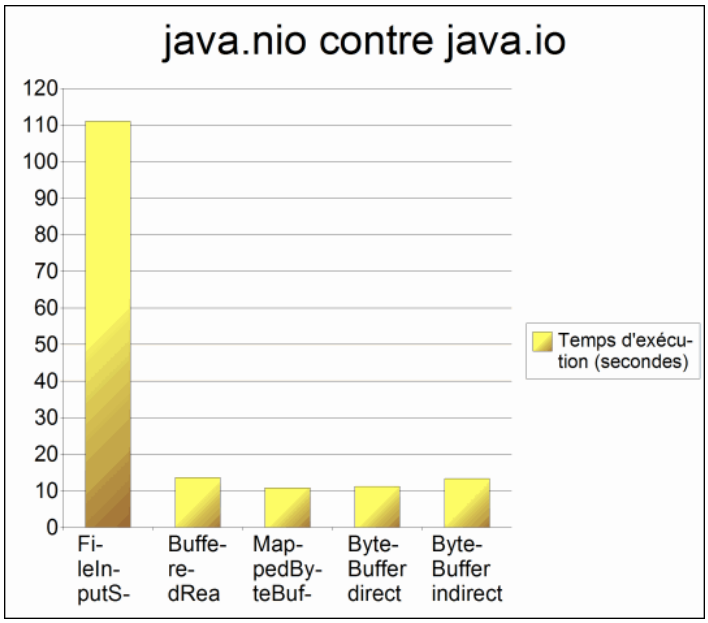
4. Buffer directs et indirects

Outre le tampon de mappage en mémoire, le nouveau paquetage propose des tampons directs ou indirects comme par exemple ByteBuffer ou FloatBuffer. Lorsque le développeur utilise un tampon direct, la machine virtuelle s'efforce de réaliser des opérations d'E/S natives directement dessus, sans passer par un tampon intermédiaire, spécifique à la JVM. Malgré leur rapidité d'exécution en règle générale, leur utilisation ne se justifie que dans le cas de manipulation de gros fichiers. En effet, le ramasse-miettes peut éprouver des difficultés à nettoyer ces objets et leur prolifération est vivement déconseillée. La création d'un tampon direct se révèle simple :

```
ByteBuffer bytes = ByteBuffer.allocateDirect(1024);
FileChannel in = new
FileInputStream("fichier").getChannel();
int read = in.read(bytes);
char c = (char) bytes.get();
```

Pour créer un tampon indirect, il suffit d'appeler la méthode allocate() à la place de allocateDirect(). Ceux-ci font moins appel à des opérations purement natives du système. Toujours dans l'exemple de notre utilitaire de comptage de lignes, nous constatons que les tampons de type direct diminuent un petit peu les performances au regard de celles obtenues par le MappedByteBuffer. Le temps d'exécution est alors de 11 secondes. Enfin, les tampons indirects donnent un temps équivalent à celui obtenu lors de l'emploi d'un BufferedReader, soit 13 secondes.

Ces divers tests montrent bien que l'optimisation d'un programme écrit en Java ne demande pas nécessairement de grands efforts. La nouvelle API NIO du JDK 1.4, réellement simple d'emploi, autorise des gains de performances non négligeables et ce pour un coût de développement quasiment nul. Ne vous fiez donc plus aux personnes clamant que Java souffre de lenteur mais apprenez-leur plutôt à utiliser correctement son API.



Retrouvez l'article en ligne de Romain Guy : Lien1

Inversion de contrôle en Java

Depuis l'avènement de la programmation orientée objet, les développeurs imaginent et implémentent des composants logiciels réutilisables. Les différentes techniques employées jusqu'à aujourd'hui ne sont malheureusement pas toujours parfaites.

1. Introduction

L'univers Java voit souvent apparaître de nouvelles technologies favorisant la conception d'architectures logicielles. Depuis quelques années, de nombreux développeurs s'intéressent à l'inversion de contrôle, ou IoC pour Inversion of Control. L'inversion de contrôle figure une nouvelle approche de la programmation de services et de composants. Pour utiliser cette technique, vous devez avoir recours à un conteneur d'inversion de contrôle comme Hivemind, PicoContainer ou Spring. Un conteneur d'IoC peut être identifié par trois caractéristiques majeures : il contient des objets, il contrôle la création de ces objets et il résout les dépendances entre les objets. De par sa nature le conteneur gère le cycle de vie de ces objets. Vous n'avez pas à créer les instances ni à libérer les ressources. Avant de nous intéresser à des exemples d'utilisation de ces conteneurs, nous allons voir l'intérêt de l'inversion de contrôle par rapport aux solutions existantes.

2. Conception par abstraction

Pour comprendre l'inversion de contrôle, nous allons prendre l'exemple d'un composant permettant de trouver des livres dans une bibliothèque. L'utilisateur pourra par exemple demander tous les livres écrits par John Grisham. Le terme composant désigne, en programmation orientée objet, un ensemble de classes formant une entité indépendante et réutilisable sans en modifier le code source. Nous utiliserons également le terme service, qui désigne un composant distant, manipulé par un système de messages ou par RPC par exemple. Notre composant bibliothèque comprendra donc au minimum une classe Bookshelf contenant pour sa part la méthode getBooksByAuthor(String author) comme le montre le listing 1. Le code présenté n'est malheureusement pas très souple puisqu'il repose sur la seule implémentation de la méthode readBooks() pour lire une liste de livres. Nous aimerions que notre bibliothèque intelligente puisse rechercher des livres dans une source de données quelconque, que ce soit un fichier CSV ou un fichier XML par exemple. Il suffit pour cela de remplacer l'appel à readBooks() par la ligne suivante :

```
List<Book> books = importer.readBooks();
```

Dans ce cas, nous invoquons la méthode readBooks() de l'objet importé qui est une instance de l'interface IBookImporter décrite dans le listing 2. Puisque nous utilisons à présent une interface pour s'abstraire de la source de données, la bibliothèque doit créer une instance d'une classe concrète, par exemple XMLBookImporter. Cette dernière classe est décrite dans le listing 3. Une manière naïve de créer l'instance serait de procéder de cette manière :

```
public Bookshelf(String bookshelf) {
  this.importer = new XMLBookImporter(bookshelf);
```

```
}
```

Cette solution fonctionne mais ne permet pas de substituer aisément l'import de livres depuis un fichier au format CSV, ou autre, à l'import XML. Pour pallier ce problème il est possible d'utiliser un singleton avec la classe DefaultBookImporter définie dans le listing 4 :

```
this.importer = DefaultBookImporter.instance();
```

Le choix d'un singleton permet à présent de modifier l'implémentation de l'import de livres de manière générale. Malheureusement, ce système ne permet d'utiliser qu'une seule implémentation à la fois. Cela peut poser des problèmes si votre application permet de créer plusieurs bibliothèques provenant de sources différentes. Pour contourner cet écueil, les développeurs et concepteurs utilisent généralement un design pattern appelé fabrique. Le listing 5 propose un exemple de fabrique pour notre exemple que nous pourrions utiliser de cette manière :

```
this.importer =
BookImporterFactory.getImporter(sourceName);
```

La fabrique proposée en exemple analyse le nom de la source de données pour déterminer la meilleure classe concrète à utiliser. L'utilisation d'une fabrique constitue une amélioration considérable par rapport au singleton mais certains problèmes subsistent. Il est notamment envisageable que la seule analyse du nom de la source ne suffise pas à définir l'implémentation adéquate. Votre composant souffrira alors de limitations qui pourront conduire à la modification de son code lorsqu'il sera réutilisé dans un autre projet. La fabrique impose également au développeur de gérer lui-même toutes les dépendances entre les objets. Une dernière solution pour découpler l'interface de programmation de l'implémentation consiste enfin à utiliser un service de recherche comme JNDI (Java Naming and Directory Interface) :

```
this.importer = (IBookImporter)
context.lookup("bookImporter");
```

Un tel mécanisme de recherche nécessite malheureusement de truffer votre code de chaînes de caractères "magiques", utilisées pour identifier les objets à rechercher. Pour changer d'objet vous devrez donc rechercher une chaîne de caractère particulière dans tous vos fichiers sources ou utiliser une classe contenant des constantes relatives au mécanisme de recherche. Cette solution, comme les précédentes, n'est donc pas totalement satisfaisante.

3. L'inversion de contrôle

Comme nous le savons à présent, les conteneurs d'inversion de contrôle prennent en charge le cycle de vie des objets ainsi que leurs dépendances. Avant de continuer, nous allons donc modifier notre bibliothèque pour faire apparaître clairement sa dépendance vis-à-vis d'IBookImporter. Son constructeur devient donc le suivant :

```
public Bookshelf(IBookImporter importer)
```

Nous pouvons à présent créer un conteneur d'IoC, enregistrer nos services auprès de celui-ci puis demander une référence au service qui nous intéresse. Pour nous la donner, le conteneur créera les instances de la classe appropriée ainsi que celles de ses dépendances. Dans notre exemple, nous demanderons le service Bookshelf. Le conteneur essayera alors de créer son instance avant de constater qu'elle dépend d'IBookImporter. Il va donc rechercher une référence à IBookImporter parmi ses services puis l'injecter dans Bookshelf. Cette injection peut être réalisée par appel du constructeur ou du mutateur approprié. Cela signifie concrètement que l'ajout de dépendances dans votre architecture ne vous demandera aucun effort. Par exemple, si vous décidez que votre bibliothèque doit également dépendre d'un IAddressBook pour gérer les emprunteurs des livres, il vous suffira de modifier le constructeur de Bookshelf :

```
public Bookshelf(IBookImporter importer, IAddressBook
addressBook)
```

A condition d'enregistrer le service IAddressBook auprès du conteneur d'IoC, vous n'aurez absolument aucun changement à apporter à votre code source. La principale difficulté de l'inversion de contrôle consiste à choisir un conteneur adapté à vos besoins. Chacun possède des caractéristiques particulières mais ils diffèrent principalement par le type d'inversion de contrôle géré, il en existe trois, et par la méthode de configuration. Les types d'IoC sont appelés Type 1 (injection d'interface), Type 2 (injection par mutateur) et Type 3 (injection par constructeur). Nous ne nous intéresserons qu'aux Type 2 et 3, les plus répandus. La configuration des services et des dépendances peut quant à elle se faire par le code ou par l'entremise d'un fichier de configuration, généralement du XML. Malgré les débats enflammés à ce sujet, il n'existe pas de type d'IoC ni de méthode de configuration supérieur aux autres, il s'agit avant tout d'une question de goût personnel.

4. Injection par mutateur avec Spring

Spring est un framework généraliste principalement utilisé pour le développement d'applications J2EE. Il comprend par exemple des couches de transaction, de persistance ou d'abstraction pour JDBC. Spring comprend notamment un conteneur d'inversion de contrôle appelé ApplicationContext. Fidèle à ses origines J2EE, ce framework utilise la terminologie des JavaBeans et des EJB plutôt que des composants et des services. Il gère enfin les IoC de Type 2 et de Type 3 et permet de configurer les dépendances non seulement avec du code mais également avec des fichiers de configuration. Leur utilisation conjointe avec une IoC de Type 2 semble être le choix favori des auteurs de Spring. Pour utiliser Spring IoC, vous devez tout d'abord récupérer les archives spring.jar et spring-context.jar de la distribution standard. Ces deux bibliothèques nécessitent en outre log4j-1.2.9.jar et commons-logging.jar que vous trouverez dans le dossier lib/ de Spring. Vous devez enfin importer les packages org.springframework.context et org.springframework.context.support dans votre code. La création et l'utilisation du conteneur est alors très simple :

```
ApplicationContext context = new
FileSystemXmlApplicationContext("spring-conf.xml");
Bookshelf shelf = (Bookshelf)
context.getBean("Bookshelf");
```

Dans cet exemple, nous utilisons la configuration contenue dans le fichier spring-conf.xml, qui se trouve dans le listing 7. Chaque composant, ou service, que nous souhaitons définir dans le conteneur est déclaré ici comme un bean. L'attribut id désigne la clé caractéristique du bean utilisée pour en obtenir une instance à partir d'un ApplicationContext. Bien que Spring gère l'injection par constructeur, nous utilisons ici l'injection par mutateur avec la balise <property> dont le contenu peut être une référence à un autre composant. Cette méthode impose de déclarer soi-même l'ensemble des dépendances ce qui peut devenir rapidement fastidieux. Fort heureusement, Sprint permet de créer les dépendances automatiquement grâce à l'attribut autowire :

```
<bean id="Bookshelf" class="Bookshelf"
autowire="byType" />
```

Les beans possèdent de nombreux attributs que vous pouvez manipuler pour gérer leur cycle de vie, leurs parents, ainsi que d'autres propriétés avancées. Vous pouvez par exemple utiliser singleton="false" pour qu'une nouvelle instance du bean soit renvoyée à chaque appel de getBean(). Spring est un excellent conteneur d'IoC qui trouvera parfaitement sa place dans vos projets J2EE.

5. Injection par constructeur avec PicoContainer

PicoContainer est un conteneur d'IoC très léger, moins de 80 ko, capable de gérer les Type 2 et 3 ainsi que de résoudre les dépendances cycliques ou en graphes. Il propose en outre un support partiel du cycle de vie des objets à travers les interfaces Startable et Disposable. Par exemple, en implémentant Disposable et sa méthode dispose() vous pourrez savoir à quel moment votre objet est libéré par PicoContainer. Vous pouvez également vous intéresser à NanoContainer, une version enrichie de ce conteneur. Son utilisation est extrêmement simple ainsi qu'en témoigne le listing 6. La méthode configureContainer() crée un nouveau conteneur en utilisant l'implémentation par défaut. Nous enregistrons ensuite nos deux services, Bookshelf et IBookImporter, en invoquant la méthode registerComponentImplementation(). Pour enregistrer un service simple comme Bookshelf, il suffit de donner la classe du service. Le cas d'IBookImporter est un peu plus compliqué puisque nous devons enregistrer des implémentations de l'interface et non l'interface elle-même. Dans ce cas, le premier paramètre définit une clé identifiant le composant de manière unique, tandis que le deuxième paramètre désigne la classe concrète à utiliser. Enfin, le dernier paramètre permet de passer des données au constructeur, comme ici le nom du fichier XML à ouvrir.

Les conteneurs d'inversion de contrôle sont simples à utiliser et permettent de découpler efficacement votre code. Les différentes philosophies proposées, le type d'injection et la méthode de configuration, doivent être étudiées convenablement avant de faire un choix. Par exemple, une configuration de dépendances en XML est très pratique pour substituer une implémentation à une autre sans recompiler votre projet. Cette configuration rend néanmoins particulièrement difficile l'utilisation du refactoring au sein d'un IDE comme Eclipse ou IntelliJ IDEA. Quel que soit votre choix, vous pourrez créer et réutiliser des composants plus facilement et plus rapidement pour tous vos projets à venir.

listing 1
```
public class Bookshelf {
  public List<Book> readBooks() {
    // lit une liste de livres
  }

  public Book[] getBooksByAuthor(String author) {
    List<Book> books = readBooks();
    List<Book> results = new ArrayList<Book>();
    for (Book b: books) {
      if (author.equals(b.getAuthor())) {
        results.add(b);
      }
    }
    return (Book[]) results.toArray(new
Book[results.size()]);
  }
}
```

listing 2
```
public interface IBookImporter {
  public List<Book> readBooks();
}
```

listing 3
```
public class XMLBookImporter implements IBookImporter {
  private String sourceName;

  public XMLBookImporter(String sourceName) {
    this.sourceName = sourceName;
  }

  public List<Book> readBooks() {
    // parcourir le fichier XML
  }
}
```

listing 4
```
public class DefaultBookImporter {
  private static IBookImporter importer = new
XMLBookImporter();

  public static IBookImport instance() {
    return importer;
  }
}
```

listing 5
```
public class BookImporterFactory {
  public static IBookImporter getImporter(String
sourceName) {
    IBookImporter importer;
    if (sourceName.startsWith("jdbc:")) {
      importer = new JDBCBookImporter(sourceName);
    } else if (sourceName.endsWith(".xml")) {
      importer = new XMLBookImporter(sourceName);
    } else {
      importer = new CSVBookImporter(sourceName);
    }
    return importer;
  }
}
```

listing 6
```
public class BookshelfTest {
  private MutablePicoContainer configureContainer() {
    MutablePicoContainer pico = new
DefaultPicoContainer();
    Parameter[] importerParams = { new
ConstantParameter("books.xml") };
    pico.registerComponentImplementation(IBookImporter.
class, XMLBookImporter.class, importerParams);
    pico.registerComponentImplementation(Bookshelf.clas
s);
    return pico;
  }

  public void test() {
    MutablePicoContainer pico = configureContainer();
    Bookshelf shelf = (Bookshelf)
pico.getComponentInstance(Bookshelf.class);
    Book[] books = shelf.getBooksByAuthor("John
Grisham");
    for (Book b: books) {
      System.out.println(b);
    }
  }
}
```

listing 7
```
<?xml version="1.0"?>
```

```
<!DOCTYPE beans SYSTEM "spring-beans.dtd" >
<beans>
  <bean id="Bookshelf" class="Bookshelf">
    <property name="importer">
      <ref local="BookImporter" />
    </property>
  </bean>
  <bean id="BookImporter" class="XMLBookImporter">
    <property name="sourceName">
       <value>books.xml</value>
    </property>
  </bean>
</beans>
```

6. Schémas

Schéma 1. Avec une dépendance directe votre code n'est ni réutilisable ni interchangeable.

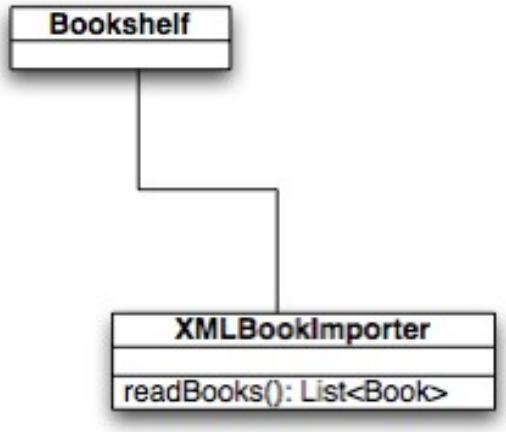


Schéma 2. L'utilisation d'une interface permet d'abstraire la dépendance et de modifier l'implémentation facilement.

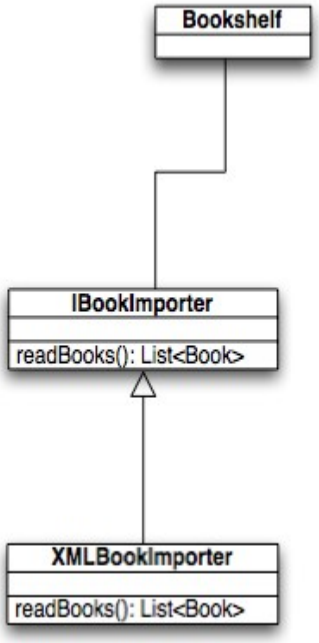


Schéma 3. Le design pattern singleton permet de modifier l'utilisation d'une implémentation globalement.

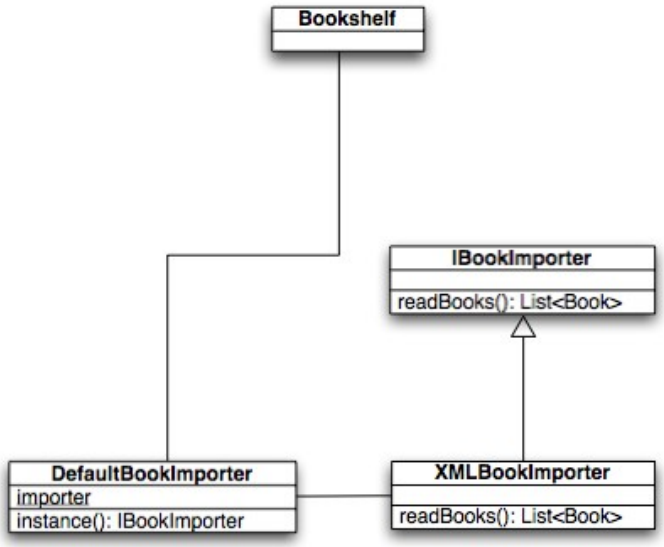


Schéma 4. La fabrique sert à choisir l'implémentation la plus appropriée à une tâche donnée.

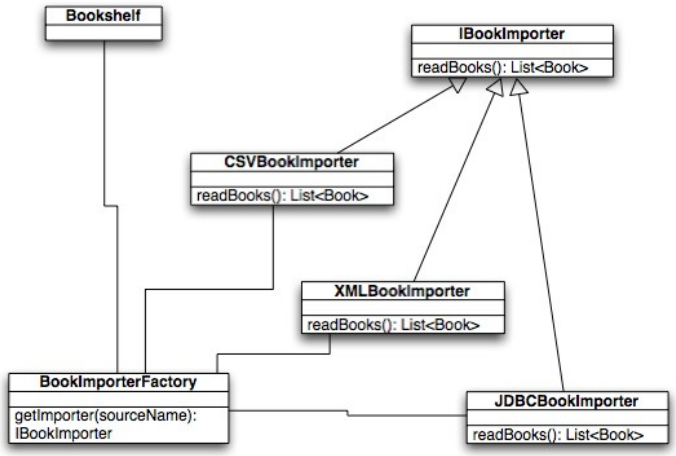
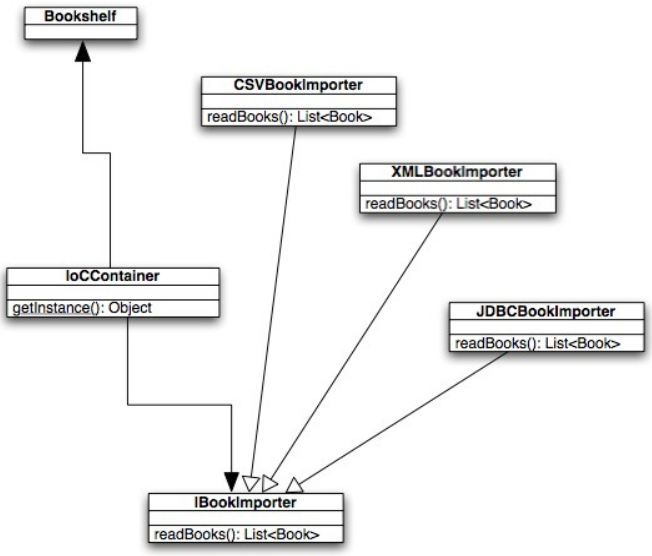


Schéma 5. Un conteneur d'IoC permet d'inverser le contrôle des dépendances et de rendre le code modulable, interchangeable et réutilisable.



Retrouvez l'article de Romain Guy en ligne : Lien2

Java Tête la première

On sait bien ce que vous pensez. ; mais que vient faire un canard dans le chapitre sur la gestion de la mémoire ? Est-ce qu'une fille dans son bain peut vraiment illustrer la subtilité d'un sujet comme le polymorphisme ? Préparez-vous à tester votre ouverture d'esprit. " Java - Tête la première " mélange jeux, images, interviews et vous amène à être actif de différentes manières. Rapide, réel, concret et amusant. Ce livre est tout cela.
Il existe mille et une façons d'apprendre. Ce livre en a choisi une bien spécifique. Sous des dehors amusants et ludiques, cet ouvrage n'en demeure pas moins parfaitement, rigoureux. Mais la différence réside dans les jeux, les énigmes à résoudre, toutes ces images, bref, la façon dont vous allez apprendre.
Cette seconde édition de " Java - Tête la première " est une introduction à la programmation orientée objet et à java. Elle se concentre sur java 5.0, la dernière version du langage java. Celle-ci représentant une mise à jour importante de la plate-forme, nous vous proposons une nouvelle édition de " Java - Tête la première ", avec beaucoup plus de code, de jeux, etc. Les sujets suivants seront abordés : les bases du langage ; le développement orienté objet ; la manière d'écrire, de tester et de déployer des applications ; l'utilisation de la bibliothèque d'API java ; le formatage des dates et des nombre ; la gestion des exception ; le multithreading ; la programmation avec Swing ; le réseau avec RMI et les sockets.
L'approche des auteurs est concentrée sur la pédagogie. Les concepts s'appuient sur des images et sur des exercices ludiques. Les différents points n'en sont alors que mieux mémorisés. Vous allez réellement penser comme un développeur orienté objet.
Si vous voulez apprendre Java mais qu'en même temps vous aimez sortir et vous amuser, alors feuilletez ce livre. Il est fait pour vous !

Critique du livre par la rédaction (Anis Frikha)

J'ai trouvé ce livre tout simplement excellent. Les auteurs de ce livre abordent Java d'une manière très originale qui consiste à combiner images, textes et jeux, ce qui permet de démystifier bon nombre de sujets délicats, ainsi on peut y trouver, par exemple, un dialogue entre une variable locale et une variable d'instance !

Chaque chapitre est clôturé par une série d'exercices très bien faits et ordonnés par difficulté croissante. Vous pouvez donc vous retrouver à jouer le rôle du compilateur ou bien de la JVM ce qui peut être très instructif.

J'ajouterais que pour tirer pleinement parti de ce livre, il vaut mieux connaître déjà un langage de programmation et qu'il soit de préférence orienté objet (mais ce n'est pas obligatoire). Si vous faites partie de cette catégorie de gens alors ce livre constitue un très bon investissement. Dans le cas contraire, passez votre chemin.

Retrouvez ce livre sur la rubrique Java : Lien3

Développement Java sous STRUTS : Version 1.2

Cet ouvrage s'adresse à toute personne désireuse de perfectionner ses connaissances sur le développement Web avec Java. Il décrit les différents constituants de Struts ainsi que les méthodes pour développer des applications utilisant ce framework.
Le chapitre 1 permet de découvrir le développement Web avec Java.
Le chapitre 2 présente le framework Struts d'un point de vue théorique puis d'un point de vue pratique en développant une première application Struts.
La Vue du modèle MVC 2 Struts est présentée dans le chapitre 3, les ActionForms et les formulaires JSP y sont détaillés.
Le chapitre 4 présente les validations en utilisant des méthodes de la classe ActionForm et/ou les validators. Il présente également différentes méthodes de validation utilisant des formulaires constitués de plusieurs pages JSP.
Le chapitre 5 présente le contrôleur représenté par la classe ActionServlet et les classes Action.
Les balises Struts sont détaillées dans le chapitre 6.
Le chapitre 7 montre comment implémenter le modèle dans une application Struts. Différentes méthodes sont proposées, la première en utilisant des classes Java simples, la deuxième détaille l'utilisation du Design Pattern DAO, la troisième utilise le framework objet/relationnel Hibernate, la dernière les EJB et les services web.
Le chapitre 8 présente des outils open source qu'il est possible d'utiliser avec Struts : ANT, Maven, Log4J, JUnit. Il présente également la future version de Struts 1.3 ainsi que quelques technologies concurrentes présentes ou futures (JSF, Shale, MyFaces).

Critique du livre par la rédaction (Edouard Kaiser)

En reprenant les bases de la programmation web (HTML, Servlet, JSP) et les infrastructures que cela met en jeu (serveurs Web, serveur d'application et de base de données), l'auteur nous entraîne petit à petit vers la programmation web structurée par le modèle MVC sous le framework Struts. Si vous ne savez pas ce qu'est le modèle MVC, aucun souci car une partie d'un chapitre lui est consacrée afin de bien assimiler ce qu'apporte ce modèle et en quoi Struts réponds à sa mise en place.
Tout les concepts du framework sont ici clairement abordées (ActionForm, taglibs Struts, Validators, etc.) à travers des exemples concis et facilement compréhensibles.
Mais ce livre ne se contente pas simplement de décrire l'utilisation du framework et ce qui le compose, l'auteur élargit le sujet en montrant comment ce framework peut être utilisé, couplé à d'autres technologies telles que les WebServices ou les EJBs. De plus des patterns tels que le pattern DAO sont également décrits (appuyé sur Hibernate). De ce fait, cet ouvrage permet à n'importe quelle personne souhaitant se lancer dans le développement web d'obtenir une vue générale concernant les différentes couches composant une application. On appréciera également une partie d'un chapitre dédiée à des outils tels que Log4j ou ANT par exemple.
Cependant j'aurais une remarque à faire concernant les exemples en eux-mêmes du livre. Même s'il est très pratique d'avoir plusieurs petits exemples pour comprendre différents concepts, je pense qu'il aurait été bon également que le livre s'appuie sur la réalisation d'une et une seule application afin de mieux assimiler en quoi justement ce modèle permet de séparer clairement les différentes couches applicatives afin d'obtenir un résultat modulaire.

Retrouvez ce livre sur la rubrique Java : Lien4

Créer son DVD d'installation de FreeBSD

Cet article présente pas à pas les étapes pour créer son propre DVD d'installation de FreeBSD.

1. Introduction

Pourquoi créer un DVD d'installation ? Malgré le fait que le DVD a pratiquement la même apparence que le CD, la capacité de stockage n'est pas négligeable. En effet, on peut y stocker 7 fois plus de données que sur un CD (soit 4,7 Go) voire plus si le DVD est en double couche. Fini le changement de CD durant la phase d'installation.

2. Pré-requis

Les pré-requis pour mener à bien cet article sont les suivants :

1. Utilisez un système FreeBSD
2. Disposez d'une connexion Internet
3. Paquetages cdrtools et dvd+rw-tools installés

Voici ci-dessous la procédure d'installation des paquetages requis.

Installation de cdrtools :

```
make -C /usr/ports/sysutils/cdrtools/ install clean
```

Installation de dvd+rw-tools :

```
make -C /usr/ports/sysutils/dvd+rw-tools/ install clean
```

Assurez-vous de disposer de tous les éléments nécessaires.

3. Préparer le terrain

Vous allez créer l'arborescence suivante avec la commande **mkdir** :

```
/home/$USER
  | - /FreeBSD
  |     | - /iso
  |     | - /tmp
  |     | - /works
  |     | - /i386
  |     -
  -
```

$USER est le nom de votre compte.

4. Obtenir les images de CD

Pour télécharger les images de CD, Vous allez utiliser la commande **fetch** :

```
cd FreeBSD/iso/
fetch
ftp://ftp.freebsd.org/pub/FreeBSD/releases/i386/ISO-IMAGES/6.2/6.2-RELEASE-i386-disc1.iso
fetch
ftp://ftp.freebsd.org/pub/FreeBSD/releases/i386/ISO-IMAGES/6.2/6.2-RELEASE-i386-disc2.iso
fetch
ftp://ftp.freebsd.org/pub/FreeBSD/releases/i386/ISO-IMAGES/6.2/6.2-RELEASE-i386-docs.iso
```

En attendant, vous pouvez regarder deux bons films à la télévision.

5. Opérations sur images

Vous allez travailler en root :

```
su -
Password:
```

Vous allez maintenant passer à la phase d'intégration des fichiers. Elle se divise en plusieurs étapes :

1. Création d'un disque en mémoire pour l'image iso x
2. Montage du disque dans le répertoire x
3. Copie des fichiers dans le répertoire x
4. Démontage du disque en mémoire
5. Suppression du disque de la mémoire

Commencez par l'image 6.2-RELEASE-i386-docs.iso.

Création d'un disque en mémoire pour l'image iso 6.2-RELEASE-i386-docs.iso :

```
mdconfig -a -f /home/$USER/FreeBSD/iso/6.2-RELEASE-i386-docs.iso
```

Montage du disque dans un répertoire /home/$USER/FreeBSD/tmp/ :

```
mount -t cd9660 /dev/md0 /home/$USER/FreeBSD/tmp/
```

Placez-vous et copiez les fichiers dans le répertoire /home/$USER/FreeBSD/works/ :

```
cd /home/$USER/FreeBSD/works/
tar -C /home/$USER/FreeBSD/tmp/ -cf - . | tar -xf -
```

Veuillez patienter, cela peut prendre quelques minutes.

Démontage du disque en mémoire :

```
umount /home/$USER/FreeBSD/tmp/
```

Suppression du disque de la mémoire :

```
mdconfig -d -u 0
```

Vous allez répéter ces opérations pour l'image 6.2-RELEASE-i386-disc2.iso. Puis, pour la dernière image 6.2-RELEASE-i386-disc1.iso. Il vous suffira juste de changer le nom des fichiers.

La copie de fichiers est terminée. Tous les fichiers nécessaires à la création du DVD se situent dans le répertoire /home/$USER/FreeBSD/works/. Visualisez son contenu :
ls /home/$USER/FreeBSD/works/

6. Vérification et manipulation sur fichiers

Avant de créer votre image ISO, vous devez vérifier quelques

paramètres.

Vérifiez que le fichier cdrom.inf pointe correctement sur le premier volume CD :

```
cat /home/$USER/FreeBSD/works/cdrom.inf
```

Vous devez obtenir :

```
CD_VERSION = 6.2-RELEASE
CD_VOLUME = 1
```

Vous devez modifier le fichier INDEX dans le répertoire /home/$USER/FreeBSD/works/packages/INDEX. En effet, actuellement le fichier INDEX pointe toujours sur le CD n°1 et n°2. Il doit pointer uniquement sur le n°1. Dans ce cas, exécutez la commande :

```
cd /home/$USER/FreeBSD/works/packages/
cat INDEX | sed "s/|2/|1/g" > INDEX.tmp
mv INDEX.tmp INDEX
```

Il reste une étape. Vous devez supprimer le répertoire rr_moved. Pour ce faire, descendez d'un cran :

```
cd ..
rm -rf rr_moved/
```

7. Création de l'image ISO

L'avant-dernière étape consiste à créer l'image ISO. Vous allez utiliser mkisofs :

```
cd /home/$USER/FreeBSD/works/
mkisofs -v -R -J -V FreeBSD6.2 -no-emul-boot -b
boot/cdboot -o /home/$USER/FreeBSD/i386/6.2-RELEASE-
i386-dvd.iso /home/$USER/FreeBSD/works/
```

Veuillez patienter quelques minutes.

```
L'image doit se trouver dans le répertoire
/home/$USER/FreeBSD/i386/.
ls /home/$USER/FreeBSD/i386/
```

Vous devez obtenir :

```
6.2-RELEASE-i386-dvd.iso
```

8. Graver l'image ISO sur DVD

Voici la dernière étape, gravez votre image ISO sur un DVD vierge.

Veuillez insérer un DVD vierge.

Exécutez la commande :

```
growisofs -dvd-compat -Z
/dev/dvd=/home/$USER/FreeBSD/i386/6.2-RELEASE-i386-
dvd.iso
```

Veuillez patienter quelques instants pendant la phase de gravure. Un petit café ? Souhaitez-vous installer FreeBSD6.2 ? Oui ? Dans ce cas, utilisez votre DVD.

9. Conclusion

Nous voici donc arrivés à la fin de cet article sur la création d'un DVD d'installation de FreeBSD6.2.

Si vous avez des commentaires, des questions, postez sur le forum afin que la communauté en profite.

10. Liens utiles

- Le site officiel de FreeBSD : Lien5
- How to make a FreeBSD DVD : Lien6

Les pages man :

- La page du manuel make : Lien7
- La page du manuel mdconfig : Lien8
- La page du manuel mount : Lien9
- La page du manuel tar : Lien10
- La page du manuel rm : Lien11
- La page du manuel mkisofs : Lien12
- La page du manuel growisofs : Lien13

Retrouvez l'article d'Olivier Regnier en ligne : Lien14

Les livres Linux

Debian GNU/Linux 3.1 (Sarge)

Ce livre est destiné à tout technicien ou administrateur système appelé à mettre en place des serveurs et des stations de travail Linux. Fondé sur la distribution Debian Sarge 3.1, il permet d'acquérir les connaissances fondamentales à l'administration de ce système d'exploitation et de découvrir ses particularités pour assurer son bon fonctionnement dans le temps. De l'installation de la distribution à la sauvegarde des données, le lecteur découvrira également la gestion des utilisateurs, des disques et des périphériques, la mise en place de services, la surveillance du système, la planification des travaux et les particularités du noyau 2.6 fourni avec Debian Sarge. Un accent particulier sera mis tout au long de ce livre sur l'optimisation du système et la gestion de la sécurité. Toutes les commandes sont présentées de façon approfondie et sont illustrées de nombreux exemples.

Critique du livre par la rédaction (Nicolas Vallée)

A la condition d'avoir une configuration matérielle reconnue par défaut par Debian, les 370 pages de cet ouvrage nous amènent à un système stable. Vous n'y trouverez aucune aide pour des configuration moins standards, et il vous faudra trouver par vous-mêmes les drivers propriétaires.

Les gestion matérielle et logique des disques est claire, ainsi que celle des groupes et des utilisateurs. Les explications sur les types d'archives et paquets sont accessibles même à un débutant. La configuration de réseaux aurait mérité d'être plus largement développée. L'administration standard du système est claire, le principal s'y trouve. La gestion et logique des sauvegardes sont accessibles. Les journaux et leur gestion sont bien détaillés. J'ai apprécié les chapitres consacrés à la sécurité, et à l'optimisation et la compilation du noyau et des modules. Enfin, le dernier chapitre consacré à x-window aurait mérité d'être un peu plus développé.

Malheureusement, il faut souvent courir d'un chapitre à l'autre pour enfin trouver les solutions pourtant simples à mettre en oeuvre.
Par ailleurs, cet ouvrage faisant partie d'un pack sorti récemment, j'ai du mal à cerner le public visé, d'autant plus la version 4.0 devait sortir... Pour résumer, je suis un peu resté sur ma faim.

Retrouvez ce livre sur la rubrique Linux : Lien15

Administration réseau sous Linux

Administration réseau sous Linux couvre tout ce qu'il y a à connaître pour administrer un réseau sous Linux. Chaque chapitre de cet ouvrage dispense l'information nécessaire à la configuration et à l'administration d'un service spécifique, permettant ainsi à l'administrateur de mettre en place et d'optimiser le fonctionnement de son réseau. Les auteurs ont adopté une démarche progressive. Ils présentent d'abord les concepts de base relatifs au réseau (TCP/IP en particulier), puis ils expliquent la configuration d'un pare-feu et l'accounting. L'ouvrage se termine par des explications concernant des services plus complexes tels que Samba, SSH, Apache et le sans fil. Parmi les sujets abordés : Configurer des interfaces Ethernet ; Paramétrer un serveur de noms (avec BIND ou djbdns) ; L'authentification avec PPP ; Mettre en place un pare-feu ; Configurer sendmail à l'aide des macros ; Router le courrier ; Se connecter à distance via ssh ; Configurer et compiler Apache. Cette troisième édition a été remise à jour pour couvrir des services réseaux indispensables tels que Apache, Samba et OpenLDAP, de même qu'elle fournit des informations récentes sur le sans fil, IMAP et IPv6. Les néophytes comme les utilisateurs plus avancés devraient trouver dans cet ouvrage la plupart des réponses aux questions qui se posent lors de la mise en œuvre d'une configuration réseau sous Linux.

Critique du livre par la rédaction (cyberzoide)

Que les administrateurs systèmes ne s'y trompent pas : cet ouvrage ne leur est pas destiné. Le public visé sont les débutants en système Linux, en particulier les étudiants en école d'informatique désireux de se former à l'administration réseau d'un serveur Linux.

Ce livre se fait fort de présenter les services et applications réseaux essentiels sous Linux, après un rappel bien nécessaire des notions de base sur les réseaux et notamment les protocoles TCP/IP. Tout y passe : le pare-feu iptables, la configuration des cartes réseaux, le routage, les connexions sécurisées SSH ainsi que les principaux services de partage de fichier et de courrier électronique. Cette troisième édition traite de sujets nouveaux tels quels que les réseaux sans fils, IPv6, etc.

Cet ouvrage constitue une bonne introduction aux services réseaux Linux et à leur configuration. Cependant, il ne dispense en rien de la lecture des très nécessaires ouvrages spécialisés pour chacun des services dont il traite. Les deux premiers chapitres consacrés à la théorie des réseaux sont courts, bien organisés et particulièrement concis ; un vrai régal pour les débutants et constituent pour eux une approche idéale.

Il faut attendre le chapitre 4 pour entrer enfin dans le vif du sujet avec la configuration réseau de base : nom d'hôte, adresses IP des cartes réseaux et routage. Puis s'enchaînent les services et les applications de base d'un serveur Linux.

On pourra regretter l'absence de certains services importants tels que SNMP et NFS ; mais les auteurs ont du faire des choix, judicieux, pour proposer un ouvrage clair et facile à lire.

Les contraintes de sécurité du système sont présentent tout au long de cet ouvrage avec un chapitre imposant sur les pare-feux. La configuration de chaque service met en exergue les nécessités de sécurité et montre clairement les moyens d'y parvenir. Les auteurs doivent en être félicités.

Bref, ce livre est un outil de formation des étudiants en informatique tout à fait intéressant.

Retrouvez ce livre sur la rubrique Linux : Lien16

Linux Administration - Noyau 2.6

Considéré par de nombreux administrateurs Linux comme une des meilleures références en langue française sur le sujet, ce guide d'autoformation de plus de 900 pages vous permettra de maîtriser rapidement tous les aspects de l'administration d'un serveur Linux : gestion des fichiers et des utilisateurs, sauvegarde et restauration, administration réseau, sécurité, optimisation des performances, etc.

Plus de 170 exercices corrigés vous permettront de tester vos connaissances et de vous entraîner aux tâches qui constituent le quotidien d'un administrateur système et réseau.

Les auteurs ont choisi de mettre l'accent sur le mode commande, plutôt que sur les outils graphiques fournis avec telle ou telle distribution. La connaissance des fichiers et des commandes qui se cachent derrière ces outils est en effet indispensable aux administrateurs opérant dans un contexte professionnel, même si certains utilitaires comme webmin, traité en annexe, peuvent faciliter leur tâche au quotidien.

Mise à jour pour couvrir la version 2.6 du noyau Linux et augmentée de plus de 250 pages, cette quatrième édition de Linux Administration peut être utilisée aussi bien avec Red Hat et Fedora qu'avec SuSE, Mandrake ou Debian, la plupart des scripts d'administration proposés étant totalement indépendants de la distribution choisie.

Critique du livre par la rédaction (Cédric Chatelain)

Sous sa couverture un peu stricte ce livre est une véritable mine d'informations. Il est très complet en ce qui concerne les fonctionnalités d'administration Linux, les fichiers de configuration, les outils fournis avec Linux... etc etc ..... Ce livre m'a vraiment emballé et m'a déjà dépanné plusieurs fois sur des sujets que je ne connaissais que très superficiellement (et que je comprends très bien maintenant).

J'ai beaucoup apprécié d'y effectuer des recherches. Les thèmes sont bien classés, les indications sont claires et précises, faciles à lire et à mettre en pratique. Tout ceci est très bien complété pas des exemples (copies d'écran avec les enchainements de commandes à passer). De plus, les explications claires permettent de bien comprendre le fonctionnement du système.

En bref, ce livre est une aide précieuse pour comprendre le fonctionnement de Linux et progresser dans la maîtrise que l'on en a. En ce qui me concerne, c'est une des sources principales d'informations en ce qui concerne l'administration Linux.

Retrouvez ce livre sur la page livres Linux : Lien17

Débuter avec le Zend Framework (approche MVC)

I. Introduction

I-A. Préambule

Ce tutoriel a été testé sur la version 1.0.0.RC1 du Zend Framework. Il a de grandes chances de fonctionner sur des versions plus récentes mais pas sur les versions antérieures à 1.0.0.RC1 à cause de la dépendance au helper ViewRenderer.

I-B. Architecture MVC

La méthode traditionnelle pour construire une application PHP est :

```
<?php
include "common-libs.php";
include "config.php";
mysql_connect($hostname, $username, $password);
mysql_select_db($database);
?>

<?php include "header.php"; ?>
<h1>Home Page</h1>

<?php
$sql = "SELECT * FROM news";
$result = mysql_query($sql);
?>
<table>
<?php
while ($row = mysql_fetch_assoc($result)) {
?>
<tr>
        <td><?php echo $row['date_created']; ?></td>
        <td><?php echo $row['title']; ?></td>
</tr>
<?php
}
?>
</table>
<?php include "footer.php"; ?>
```

Au long du cycle de vie de l'application, ce type de code devient impossible à maintenir car le client continue de demander des modifications, qui sont codées à plusieurs endroits du code principal.

Une méthode permettant d'améliorer les possibilités de maintenance des applications est de séparer le code en différentes parties (et habituellement en différents scripts) :

- **Modèle** : La partie "modèle" de l'application est celle concernée par les détails des informations à être affichées. Dans l'exemple ci-dessus, c'est le concept de "news". Ainsi, cette partie s'occupe généralement de la "logique d'entreprise" de l'application ; elle a tendance à charger et à sauvegarder vers des bases de données.
- **Vue** : La vue contient les morceaux de l'application qui affichent les informations à l'utilisateur. C'est généralement le HTML.
- **Contrôleur** : Le Contrôleur lie ensemble le Modèle et la Vue pour s'assurer que les informations correctes sont affichées dans la page.

Le Zend Framework utilise l'architecture Modèle-Vue-Contrôleur (MVC), utilisée pour faciliter le développement et la maintenance en séparant les composants d'une application.

I-C. Matériel requis

Le Zend Framework a besoin des éléments suivants :

- PHP 5.1.4 (ou ultérieur) ;
- Un serveur Web supportant la fonctionnalité mod_rewrite (ce tutoriel suppose l'utilisation d'Apache).

I-D. Récupérer le framework

Le Zend Framework est disponible à l'adresse http://framework.zend.com/download/stable au format .zip ou .tar.gz. Au moment de la rédaction, la version 0.9 est la version actuelle. Vous devez utiliser la version 0.9 pour pouvoir profiter de ce tutoriel.

II. Organisation

II-A. Structure des répertoires

Alors que le Zend Framework n'oblige pas à utiliser une structure particulière de répertoires, la documentation en recommande une. Cette structure suppose que vous ayez un contrôle complet sur la configuration de votre serveur Apache mais, puisque nous voulons nous simplifier la vie, nous allons opérer une légère modification.

Commencez par créer un répertoire "zf-tutorial" dans le dossier racine du serveur Web. Cela signifie que l'URI pour obtenir l'application sera : http://localhost/zf-tutorial/.
Créez la structure suivante pour contenir les fichiers de l'application :

```
zf-tutorial/
        /application
                /controllers
                /models
                /views
                        /filters
                        /helpers
                        /scripts
        /library
        /public
                /images
                /scripts
                /styles
```

Comme vous pouvez le voir, nous avons des dossiers distincts pour les fichiers du Modèle, de la Vue et du Contrôleur de l'application. Les images, scripts (Javascript) et CSS sont situés dans des dossiers distincts, dans le dossier "public". Les fichiers téléchargés du Zend Framework seront placés dans le dossier "library". Si vous utilisez d'autres bibliothèques, vous devriez les y mettre également.

Extrayez l'archive, ZendFramework-0.9.1-Beta.zip dans mon cas, dans un dossier temporaire. Tous les fichiers sont placés dans un sous dossier appelé "ZendFramework-0.9.1-Beta". Copiez le contenu de "ZendFramework-0.9.1-Beta/library/Zend" dans "zf-tutorial/library/". Votre dossier "zf-tutorial/library" devrait maintenant contenir un sous dossier "zend".

II-B. Bootstrapping

II-B-1. Le concept

Le Contrôleur du Zend Framework, Zend_Controller, est prévu pour supporter des sites avec des URIs propres. Pour y parvenir, toutes les URIs doivent passer par un script unique, index.php, connu en tant que "". Cela nous fournit un point central pour toutes les pages de l'application et nous assure que l'environnement est correctement mis en place pour exécuter l'application. Nous y parvenons au moyen d'un fichier .htaccess dans le répertoire "zf-tutorial" :

```
zf-tutorial/.htaccess
RewriteEngine on
RewriteRule .* index.php

php_flag magic_quotes_gpc off
php_flag register_globals off
```

La commande RewriteRule est vraiment simple et peu être interprêtée comme "pour toute URI, utilise index.php".

Nous pouvons également mettre en place quelques paramètres php.ini pour la sécurité. Ils devraient déjà être corrects mais nous voulons en être certains !

La directive php_flag ne fonctionne que si vous utilisez mod_php. Si vous utilisez fast-CGI, alors vous devez vérifier que votre php.ini est correct.

Cependant, les images, scripts et CSS ne doivent pas passer par index.php : en les plaçant dans "public", nous pouvons aisément configurer Apache au moyen d'un autre .htaccess afin d'envoyer directement ces fichiers :

```
zf-tutorial/public/.htaccess
RewriteEngine off
```

Bien que cela ne soit pas strictement nécessaire sans règles de réécriture locales, nous pouvons ajouter quelques .htaccess supplémentaires pour nous assurer que les répertoires "application" et "library" sont protégés :
zf-tutorial/application/.htaccess
deny from all

```
zf-tutorial/library/.htaccess
deny from all
```

Pour que les .htaccess fonctionnent avec Apache, il faut que la directive de configuration AllowOverride soit mise à All dans votre fichier httpd.conf. L'idée des multiples fichiers .htaccess vient de l'article de Jayson Minard " Blueprint for PHP Applications: Bootstrapping (part 2)". Je vous recommande de lire les deux cours.

II-B-2. Le script : index.php

Notre script est "**zf-tutorial/index.php**" et nous allons commencer avec le code suivant :

zf-tutorial/index.php
```
<?php
error_reporting(E_ALL|E_STRICT);
date_default_timezone_set('Europe/Paris');

set_include_path('.' . PATH_SEPARATOR . './library'
         . PATH_SEPARATOR . './application/models/'
     . PATH_SEPARATOR . get_include_path());
include "Zend/Loader.php";

Zend_Loader::loadClass('Zend_Controller_Front');

// setup controller
$frontController =
Zend_Controller_Front::getInstance();
$frontController->throwExceptions(true);
$frontController-
>setControllerDirectory('./application/controllers');

// run!
$frontController->dispatch();
```

Nous ne mettons pas de ?> à la fin du script puisque ce n'est pas nécessaire et que cela peut donner lieu à des erreurs difficiles à identifier en cas d'utilisation de la fonction **header()**, en cas d'espaces additionnels après cette balise.

Étudions maintenant ce script.
```
error_reporting(E_ALL|E_STRICT);
date_default_timezone_set('Europe/Paris');
```

Si **display_errors=on** dans votre php.ini, alors ces lignes vous assurent que vous verrez les erreurs à l'écran. Nous précisons également notre zone temporelle, tel qu'il est requis depuis PHP 5.1+ : évidemment, il faut mettre ici votre propre zone.

```
set_include_path('.' . PATH_SEPARATOR . './library'
         . PATH_SEPARATOR . './application/models/'
     . PATH_SEPARATOR . get_include_path());
include "Zend/Loader.php";
```

Le Zend Framework est constitué tel que ses scripts doivent être dans l'include path de PHP. Nous y mettons également les Modèles afin de pouvoir charger plus facilement nos classes par la suite. Pour démarrer, nous avons besoin du script Zend/Loader.php pour nous donner accès à la classe Zend_Loader, qui dispose des méthodes statiques nécessaires au chargement de toute autre classe du Zend Framework.

```
Zend_Loader::loadClass('Zend_Controller_Front');
```

Zend_Loader::loadClass charge la classe en paramètre. Le procédé est de remplacer les caractères "_" du nom de la classe par des sépérateurs de répertoire, puis d'ajouter ".php". Ainsi, la classe "Zend_Controller_Front" est chargée depuis le script "Zend/Controller/Front.php". Si vous suivez la même convention pour vos classes, vous pourrez également utiliser la méthode **Zend_Loader::loadClass()** pour les charger. La première classe dont nous avons besoin est le contrôleur primaire.

Le contrôleur primaire est une classe de routage pour lier l'URI demandée et la fonction PHP correcte permettant d'afficher la page. Afin que le routeur puisse agir, il doit découvrir la portion de l'URI qui désigne l'index.php afin de pouvoir déterminer quels sont les éléments de l'URI à partir de là. C'est effectué par un objet Request, qui s'arrange très bien pour deviner l'URI fondamentale. Cependant, si cela ne fonctionne pas dans votre environneent, vous pouvez la surcharger avec la méthode **$frontController->setBaseUrl()**.

Nous devons configurer le contrôleur primaire afin de trouver où trouver les contrôleurs :

```
$frontController =
Zend_Controller_Front::getInstance();
$frontController-
>setControllerDirectory('./application/controllers');
$frontController->throwExceptions(true);
```

Puisque ceci est un tutoriel, nous utilisons un système de test : j'ai donc décidé de demander au contrôleur primaire de lancer toutes les exceptions qui peuvent survenir. Par défaut, le contrôleur primaire les attrape toutes à notre place et les enregistre dans la propriété "_exceptions" de l'objet Response créé. Cet objet contient toutes les informations sur la réponse à l'URI demandée. Le contrôleur primaire va automatiquement envoyer les en têtes et afficher le contenu de la page juste avant de terminer son travail.

Cela peut être assez déroutant pour les développeurs qui découvrent le framework, ainsi il est plus facile pour vous de relancer de manière à rendre plus visibles les exceptions. Bien entendu, en environnement de production, il ne faudrait de toute manière pas afficher les erreurs à l'utilisateur !

Nous arrivons finalement au cœur de la question et nous exécutons l'application :

```
// run!
$frontController->dispatch();
```

Si vous allez à http://localhost/zf-tutorial/ pour essayer, vous devriez voir quelque chose de similaire à :

```
Fatal error: Uncaught exception
'Zend_Controller_Dispatcher_Exception' with message
'Invalid controller specified (index)' in…
```

Cela nous informe que nous n'avons pas encore mis en place notre application. Avant de pouvoir le faire, nous devrions étudier ce que nous allons programmer, concentrons-nous donc là-dessus dès à présent.

II-C. Le site Web

II-C-1. Thème du site

Nous allons construire une liste très simple de notre collection de CDs. La page principale va nous permettre d'afficher la liste et d'ajouter, de modifier ou de supprimer des disques. Nous allons enregistrer notre liste dans une base de données au schéma suivant :

Champ	Type	NULL	Commentaires
Id	Integer	Non	Clef primaire, auto incrément
Artist	Varchar(100)	Non	
Title	Varchar(100)	Non	

II-C-2. Pages requises

Les pages suivantes seront nécessaires :

- Page d'accueil : Cela affichera la liste des disques, fournira des liens pour les modifier et supprimer ainsi que pour en ajouter ;
- Ajouter un album : Cette page proposera un formulaire permettant d'ajouter un disque ;
- Modifier un album : Cette page proposera un formulaire permettant de modifier un disque ;
- Supprimer un album : Cette page confirmera que nous souhaitons supprimer un album, puis le supprimera.

II-C-3. Organiser les pages

Avant de mettre en place les scripts, il faut comprendre comment le framework s'attend à ce que les pages soient organisées. Chaque page de l'application est connue comme une "action" et les actions sont regroupées en "contrôleurs". Par exemple pour une URI du format "http://localhost/zf-tutorial/actualités/voir", le contrôleur est "actualités" et l'action est "voir". Cela permet de regrouper les actions en relation. Par exemple, un contrôleur "actualités" peut avoir les actions "récentes", "archives" et "voir". Le système MVC du Zend Framework supporte également le regroupement de contrôleurs mais notre application n'est pas suffisamment conséquente pour qu'il soit nécessaire de s'en préoccuper !

Le contrôleur du Zend Framework réserve une action "index" comme action par défaut. C'est-à-dire que pour l'URI "http://localhost/zf-tutorial/actualités/", l'action "index" est exécutée. Le framework réserve également un nom de contrôleur si aucun n'est fourni dans l'URI : aucune surprise qu'il soit également appelé "index". Ainsi, l'URI "http://localhost/zf-tutorial/" appelle le contrôleur "index" avec l'action "index".

Puisque c'est un cours simple, nous n'allons pas nous compliquer avec des choses "complexes" comme une séquence de connexion ! Cela attendra un prochain tutoriel...

Puisque nous avons quatre actions à appliquer à tous les albums, nous allons les regrouper en un seul contrôleur comme quatre actions. Nous utiliserons le contrôleur par défaut et les actions sont :

Page	Controleur	Action
Accueil	index	index
Ajouter	index	ajouter
Modifier	index	modifier
Supprimmer	index	supprimer

Simple, non ?

III. Le Contrôleur

III-A. Mise en place du Contrôleur

Nous sommes maintenant prêts à mettre en place le contrôleur. Dans le Zend Framework, le contrôleur est une classe qui doit être appelée "**{Nom du contrôleur}Controller**". **{Nom du contrôleur}** doit commencer par une lettre majuscule.

Cette classe doit être dans un script appelé **{Nom du contrôleur}Controller.php** dans le répertoire du contrôleur spécifié. De nouveau, **{Nom du contrôleur}** doit commencer par

une lettre majuscule et ne contenir que des minuscules par la suite. Chaque action est une fonction publique dans le contrôleur et doit être appelée **{nom de l'action}Action**. Dans ce cas, {nom de l'action} doit commencer par une lettre minuscule.

Notre contrôleur est donc nommé IndexController et défini dans "**zf-tutorial/application/controllers/IndexController.php**" :

zf-tutorial/application/controllers/IndexController.php
```
<?php

class IndexController extends Zend_Controller_Action {
    function indexAction(){
        echo "<p>dans
IndexController::indexAction()</p>";
    }

    function ajouterAction(){
        echo "<p>dans
IndexController::ajouterAction()</p>";
    }

    function modifierAction(){
        echo "<p>dans
IndexController::modifierAction()</p>";
    }

    function supprimerAction(){
        echo "<p>dans
IndexController::supprimerAction()</p>";
    }
}
```

Initialement, nous l'avons défini afin que chaque action affiche son nom. Essayez cela en allant aux adresses suivantes :

URI	Texte affiché
http://localhost/zf-tutorial/	dans IndexController::indexAction()
http://localhost/zf-tutorial/index/ajouter	dans IndexController::ajouterAction()
http://localhost/zf-tutorial/index/modifier	dans IndexController::modifierAction()
http://localhost/zf-tutorial/index/supprimer	dans IndexController::supprimerAction()

Note du traducteur : J'ai traduit les noms des actions afin d'obtenir des URIs propres et en français, mais on voit facilement que les méthodes portent des noms bien malheureux. On a par exemple l'impression de vouloir "supprimer une action" alors qu'il s'agit de "l'action supprimer".

Nous avons maintenant un routeur correct et l'action correcte est exécutée pour chaque page de l'application. Si cela ne fonctionne pas pour vous, rendez-vous à la partie "Résolution de problèmes" vers la fin de ce cours.

Il est temps de construire la Vue.

IV. La Vue (les gabarits)

IV-A. Mise en place de la Vue

Le composant Vue du Zend Framework, sans surprise, est nommé Zend_View. Ce composant nous aidera à séparer le code d'affichage du code des méthodes d'action.

L'usage fondamental de Zend_View est :
```
$view = new Zend_View();
$view->setScriptPath('/path/to/view_files');
echo $view->render('view.php');
```

Il est évident que si nous devions utiliser ce squelette dans chacune de nos méthodes d'action, nous serions en train de répéter le code de préparation qui n'a pas d'intérêt pour l'action. Nous devrions plutôt initialiser la Vue autre part, puis accéder depuis chaque méthode d'action à notre objet déjà créé.

Les concepteurs du Zend Framework ont prévu ce type de problème, ainsi une solution est prévue pour nous dans un "action helper" (assistant d'action). **Zend_Controller_Action_Helper_ViewRenderer** créé une propriété de vue ($this->view) pour que nous puissions l'utiliser et rend également un script de vue. Pour le rendu, l'assistant demande à l'objet Zend_View de regarder dans **views/scripts/{nom du contrôleur}** afin de rechercher les scripts de Vue à rendre. Le rendement d'un tel script est réalisé par render(), qui va rendre (par défaut, du moins) le script "{controller name}.phtml" et le concaténer au corps de la réponse de l'objet Response. Cet objet est utilisé pour rassembler les en têtes, le corps et les exceptions générées comme résultat de l'utilisation du système MVC. Le contrôleur primaire envoie les headers suivi du contenu du corps à la fin de la répartition (dispatch).

Pour intégrer la Vue à notre application, nous devons initialiser la Vue avec la méthode init() puis nous assurer d'appeler render() dans chaque action. Nous devons également créer des scripts de Vue avec du code de test d'affichage.

Voici les modifications à IndexController :

zf-tutorial/application/controllers/IndexController.php
```
<?php

class IndexController extends Zend_Controller_Action {
    function indexAction(){
        $this->view->title = "Mes albums";
        $this->render();
    }

    function ajouterAction(){
        $this->view->title = "Ajouter un nouvel album";
        $this->render();
    }

    function modifierAction(){
        $this->view->title = "Modifier un album";
        $this->render();
    }

    function supprimerAction(){
        $this->view->title = "Supprimer un album";
        $this->render();
    }
}
```

Comme vous pouvez le voir, nous ajoutons une fonction init() qui

est automatiquement appelée par le constructeur de Zend_Controller_Action. Cela nous assure d'initialiser la Vue tout au début et nous pouvons être certains que c'est prêt à l'utilisation dans les méthodes d'action.

Dans chaque méthode, nous assignons une propriété $title et nous appelons render() afin d'afficher le gabarit.

L'affichage effectif n'est pas fait pour le moment, car il est pris en charge par le contrôleur primaire à la fin de la répartition.

Nous devons maintenant ajouter nos quatre scripts d'action à notre application. Ces scripts sont connus comme des "gabarits" (templates) et la méthode render() s'attend à ce que chaque gabarit s'appelle en fonction de son action et qu'il porte l'extension ".phtml" pour montrer que c'est un gabarit. Le script doit être dans un sous dossier dont le nom dépend du contrôleur, ainsi les quatre scripts sont :

zf-tutorial/application/views/scripts/index/index.phtml
```
<html>
<head>
        <title><?php echo $this->escape($this->title); ?
></title>
</head>
<body>
        <h1><?php echo $this->escape($this->title); ?
></h1>
</body>
</html>
```

zf-tutorial/application/views/scripts/index/ajouter.phtml
```
<html>
<head>
        <title><?php echo $this->escape($this->title); ?
></title>
</head>
<body>
        <h1><?php echo $this->escape($this->title); ?
></h1>
</body>
</html>
```

zf-tutorial/application/views/scripts/index/modifier.phtml
```
<html>
<head>
        <title><?php echo $this->escape($this->title); ?
></title>
</head>
<body>
        <h1><?php echo $this->escape($this->title); ?
></h1>
</body>
</html>
```

zf-tutorial/application/views/scripts/index/supprimer.phtml
```
<html>
<head>
        <title><?php echo $this->escape($this->title); ?
></title>
</head>
<body>
        <h1><?php echo $this->escape($this->title); ?
></h1>
</body>
</html>
```

Tester les quatre actions devrait afficher les titres en gras.

IV-B. Code HTML en commun

Il devient très vite évident que nous avons beaucoup de code HTML répété dans nos Vues. Nous allons mettre en facteur dans le répertoire "scripts" le code qui est commun à deux vues : "header.phtml" et "footer.phtml".

Les nouveaux fichiers sont :

zf-tutorial/application/views/scripts/header.phtml
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xml:lang="en" lang="en">
<head>
    <meta http-equiv="Content-Type"
content="text/html;charset=utf-8" />
        <title><?php echo $this->escape($this->title); ?
></title>
</head>
<body>
<div id="content">
```

zf-tutorial/application/views/scripts/footer.phtml
```
</div>
</body>
</html>
```

De nouveau, notre Vue a besoin d'être modifiée :

zf-tutorial/application/views/scripts/index/index.phtml
```
<?php echo $this->render('header.phtml'); ?>
<h1><?php echo $this->escape($this->title); ?></h1>
<?php echo $this->render('footer.phtml'); ?>
```

zf-tutorial/application/views/scripts/index/ajouter.phtml
```
<?php echo $this->render('header.phtml'); ?>
<h1><?php echo $this->escape($this->title); ?></h1>
<?php echo $this->render('footer.phtml'); ?>
```

zf-tutorial/application/views/scripts/index/modifer.phtml
```
<?php echo $this->render('header.phtml'); ?>
<h1><?php echo $this->escape($this->title); ?></h1>
<?php echo $this->render('footer.phtml'); ?>
```

zf-tutorial/application/views/scripts/index/supprimer.phtml
```
<?php echo $this->render('header.phtml'); ?>
<h1><?php echo $this->escape($this->title); ?></h1>
<?php echo $this->render('footer.phtml'); ?>
```

IV-C. Ajout de styles

Bien qu'il s'agisse d'un tutoriel simple, nous aurons besoin d'un fichier CSS pour que notre application paraisse un minimum présentable ! Cela pose en fait un problème mineur, car nous ne savons pas réellement comment référencer la CSS puisque l'URI n'indique pas le bon répertoire racine. Pour y remédier, nous utilisons la méthode getBaseUrl() qui est dans la requête et nous l'envoyons à la Vue. Cela nous donne la partie de l'URI que nous ne connaissons pas.

Modifiez IndexController::Init() de cette manière :

```
...
    function init()
    {
        $this->view->baseUrl = $this->_request-
>getBaseUrl();
```

```
    }
...
```

Nous devons ajouter la CSS à la section <head> du fichier header.phtml :

zf-tutorial/application/views/scripts/header.phtml

```
...
<head>
    <meta http-equiv="Content-Type"
content="text/html;charset=utf-8" />
        <title><?php echo $this->escape($this->title); ?
></title>
        <link rel="stylesheet" type="text/css"
media="screen"
              href="<?php echo $this-
>baseUrl;?>/public/styles/site.css" />
</head>
...
```

Enfin, le style :

zf-tutorial/public/styles/site.css

```
body,html {
        font-size:100%;
        margin: 0;
        font-family: Verdana,Arial,Helvetica,sans-serif;
        color: #000;
        background-color: #fff;
}

h1 {
        font-size:1.4em;
        color: #800000;
        background-color: transparent;
}

#content {
        width: 770px;
        margin: 0 auto;
}

label {
        width: 100px;
        display: block;
        float: left;
}

#formbutton {
        margin-left: 100px;
}

a {
        color: #800000;
}
```

Cela devrait donner un rendu un peu meilleur !

Retrouvez la suite de la traduction de l'article de Rob Allen en ligne : Lien19

Vous y apprendrez comment construire le modèle de votre application.

Vous pouvez aussi retrouvez d'autres ressources sur le framework Zend dans la rubrique Zend-Framework de Developpez.com : Lien71

Utilisez l'effet Glass de Vista dans vos applications WinForm

Windows Vista et ses fenêtres transparentes vous fait rêver ? Apprenez à utiliser le thème Aero Glass de Windows Vista pour que vous puissiez, vous aussi, avoir de la transparence dans vos fenêtres Winform.

1. Introduction

Comme vous le savez probablement tous, Windows Vista introduit avec lui un nouveau thème: Aero. Un des aspects les plus connus de ce thème est le fait que la barre de titre et les contours des fenêtres sont transparents et laissent apparaître l'arrière des fenêtres légèrement flouté.

Toutes les fenêtres des applications écrites avant ou pour Windows Vista tirerons automatiquement parti de ces nouveautés sans que le développeur n'ait besoin d'y retoucher. Cependant vous avez peut-être remarqué que certaines applications étendent cet effet de transparence à toute la zone client et non pas seulement à la barre de titre et aux contours. C'est notamment le cas du lecteur Windows Media Player dont voici une image en mode réduit:

Pas mal, non ? Et bien sachez qu'il vous est possible d'utiliser l'API qui se cache derrière tout ça pour que vous puissiez, vous aussi, étendre la transparence à toute la zone cliente de vos applications WinForm. C'est ce que nous allons voir au travers de cet article.

2. Desktop Window Manager

Le Desktop Window Manager (DWM) est la nouvelle interface qui contrôle l'affichage et le rafraîchissement des fenêtres en cours d'exécution sur le bureau Windows Vista. Le DWM contrôle la manière dont les fenêtres interagissent avec le moteur de composition de bureau. C'est grâce à lui que vous avez des fonctionnalités comme la transparence, le flip 3D, les miniatures des applications lancées, etc. Pour plus d'informations sur DWM, je vous invite à suivre les différents liens présents à la fin de cet article.

Toutes les applications que vous avez déjà programmées tirent profit du DWM sans avoir à subir de modification (comme la transparence des rebords des fenêtres). Toutefois, si vous souhaitez contrôler ou accéder aux fonctionnalités de DWM, vous devrez appeler les interfaces se trouvant dans dwmapi.dll (l'interface publique de DWM). C'est précisément ce que nous allons devoir faire pour étendre l'effet de transparence à la zone cliente d'une fenêtre.

3. Avant de commencer

Pour pouvoir utiliser l'effet de transparence, votre programme devra vérifier certaines conditions concernant son environnement.

Il faudra tout d'abord vérifier que l'application s'exécute bien sous Windows Vista. En effet, vouloir utiliser l'API de DWM sous Windows XP, par exemple, risque de vous poser quelques problèmes.

Vous pouvez donc par exemple insérer ce bout de code dans la fonction main de votre programme :

```
// on vérifie qu'on se trouve bien au moins sous Vista
if (Environment.OSVersion.Version.Major < 6){
    MessageBox.Show("Windows Vista est requis.",
"Erreur");
    return;
}
```

Ainsi, l'application se fermera si elle ne se trouve pas sous Vista. Cela peut être ennuyeux si vous souhaitez tout de même la faire tourner sous Windows XP par exemple. Pour contourner ce problème vous pouvez, par exemple, faire la vérification de la version de l'OS non pas dans la fonction main mais aux endroits où vous utilisez l'API de DWM. Vous activerez ainsi la transparence seulement si vous êtes sous Vista et pas dans les autres cas.

Un autre point à vérifier est l'activation ou non d'Aero. En effet, un utilisateur sous Windows Vista peut ne pas l'avoir activé (ou sa configuration matérielle ne lui permet pas de le faire). Cette vérification peut se faire en appelant la fonction DwmIsCompositionEnabled de l'API de DWM qui permet obtenir l'état de composition DWM du bureau. Nous verrons plus loin comment utiliser cette fonction.

4. L'API de DWM

C'est maintenant que les problèmes commencent.En effet, nous souhaitons utiliser la dll dwmapi.dll dans notre programme C#. Or cette dernière n'a pas été écrite en .NET, elle est en code dit non géré (ou non managé). Pour pallier ce problème, on utilise le mécanisme de : P/Invoke (Platform Invoke). Le site MSDN nous donne une rapide définition de ce mécanisme:

P/Invoke est l'abréviation de Platform Invoke et offre les fonctionnalités permettant d'accéder aux fonctions, structures et rappels dans les DLL non gérées. P/Invoke offre une couche de traduction permettant d'aider les développeurs en les autorisant à étendre la bibliothèque des fonctionnalités disponibles, au-delà de la bibliothèque gérée de la BCL (Base Class Library) du .NET Framework.

Pour plus d'informations sur ce mécanisme, je vous invite à aller consulter cet article (Lien20) de Thomas Lebrun ou encore celui-ci (Lien21) sur le site MSDN.

Vous y avez jeté un oeil ? Parfait, nous allons pouvoir continuer. Vous avez donc compris que nous allons devoir créer des wrappers pour les fonctions et structures requises par le DWM afin que nous puissions les appeler à partir de notre programme C#. Une description de l'API de DWM est disponible sur MSDN à cette adresse: Lien22

5. Vérifier l'activation de la composition

La première chose est donc de vérifier l'activation de la composition sous Vista en utilisant la fonction **DwmIsCompositionEnabled** dont voici la signature non gérée :

```
HRESULT DwmIsCompositionEnabled(
    BOOL *pfEnabled );
```

Nous déclarons le wrapper de la fonction P/Invoke gérée de la façon suivante :

```
[DllImport("dwmapi.dll", PreserveSig = false)]
public static extern bool DwmIsCompositionEnabled();
```

Si vous voulez en savoir plus sur le tag **PreserveSig**, allez jetez un coup d'oil sur cette page MSDN (Lien23). Sinon, sachez simplement qu'il permet de transformer la valeur de retour **HRESULT** directement en exception si besoin.

Ce wrapper n'est que le premier d'une longue liste. Pour une meilleure lisibilité vous devriez les déclarer dans une classe à part. Vous pouvez vous inspirer des sources données en exemple. A l'intérieur se trouve la classe **WrappersDWM** qui référence tous les wrappers déclarés.

N'oubliez pas de référencer l'espace de noms **System.Runtime.InteropServices** en utilisant la directive suivante:

```
using System.Runtime.InteropServices;
```

Une fois cela terminé, vous serez en mesure d'appeler la fonction **DwmIsCompositionEnabled** comme s'il s'agissait d'une fonction managée.

Voici un exemple de code que vous pouvez utiliser pour traiter le cas ou la composition est active et le cas où elle ne l'est pas:

```
//on vérifie si la composition est activée ou non
if (WrappersDWM.DwmIsCompositionEnabled()){
    //on peut activer la transparence
} else {
    //pas de transparence ici !
}
```

Nous sommes maintenant capables de savoir si la composition est activée ou non. Mais il reste un petit problème. En effet, l'utilisateur peut à tout moment désactiver Aero. Votre application doit être capable de réagir immédiatement (si besoin) à cette modification. Heureusement, lorsque l'état de composition du bureau est modifié, un message système **WM_DWMCOMPOSITIONCHANGED** est diffusé. Nous pouvons donc le récupérer et effectuer les traitements nécessaires si besoin. Par contre, ce message ne nous informe pas sur l'état (activé ou non) de la composition. Vous devrez refaire un appel à la méthode DwmIsCompositionEnabled pour le déterminer.

Pour récupérer les messages Windows nous devons redéfinir la méthode **WndProc**. Cette technique n'est pas propre à Vista, elle existe depuis la version 1.0 du FrameWork. Si vous souhaitez plus d'informations sur le sujet rendez-vous à l'adresse suivante: Lien24.

Voici à quoi pourrait ressembler le code de cette fonction:

```
protected override void WndProc(ref Message msg){
    base.WndProc(ref msg);

    const int WM_DWMCOMPOSITIONCHANGED = 0x031E;
//valeur associée au message

    switch (msg.Msg){
        case WM_DWMCOMPOSITIONCHANGED:
            if (WrappersDWM.DwmIsCompositionEnabled()){
                MessageBox.Show("Composition activée");
                //on peut activer la transparence
            } else {
                MessageBox.Show("Composition non
activée");
                //pas de transparence ici !
            }
            break;
    }
}
```

Il existe d'autres messages système liés à DWM qu'il peut être intéressant d'écouter mais que nous ne détaillerons pas ici:

- **WM_DWMCOLORIZATIONCOLORCHANGED** est envoyé lorsque la couleur ou l'opacité de l'effet de verre est modifiée. Les paramètres vous informent de la nouvelle couleur et de la nouvelle opacité.
- **WM_DWMNCRENDERINGCHANGED** est envoyé lorsque le rendu DWM change sur la zone non client.
- **WM_DWMWINDOWMAXIMIZEDCHANGE** est envoyé lorsqu'une fenêtre compositée DWM est agrandie ou minimisée. Par exemple, la barre des tâches réagit à cet évènement en devenant opaque.

6.1. Première technique

Nous allons ici utiliser la fonction DwmExtendFrameIntoClientArea dont voici la signature non gérée:

```
HRESULT DwmExtendFrameIntoClientArea(
    HWND hWnd,     const MARGINS *pMarInset );
```

Et le wrapper associé:

```
[DllImport("dwmapi.dll", PreserveSig = false)]
public static extern void
DwmExtendFrameIntoClientArea(IntPtr hWnd, ref MARGINS
pMargins);
```

Cette fonction prend en paramètre le handle de la fenêtre dont les bords doivent être étendus ainsi qu'une structure MARGINS qui décrit comment les quatre marges de la fenêtre doivent être étendues.

Voici la version non gérée de cette structure:

```
typedef struct _MARGINS {
int cxLeftWidth;
int cxRightWidth;
```

```
int cyTopHeight;
int cyBottomHeight;
} MARGINS, *PMARGINS;
```

Et sa version gérée:

```
[StructLayout(LayoutKind.Sequential)]
public struct MARGINS{
    public int cxLeftWidth, cxRightWidth, cyTopHeight,
cyBottomHeight;

    public MARGINS(int left, int right, int top, int
bottom){
        cxLeftWidth = left; cyTopHeight = top;
        cxRightWidth = right; cyBottomHeight = bottom;
    }
}
```

La première chose à faire est donc de construire un objet MARGINS qui permet d'indiquer pour chaque marge la distance (en pixels) à laquelle elle doit s'étendre à l'intérieur de la zone cliente. Une valeur de -1 indique que la marge doit s'étendre sur toute la zone cliente. Pour annuler l'extention de la marge il suffit de redéfinir un objet MARGINS avec des valeurs à 0.
Il faut ensuite appeler la méthode DwmExtendFrameIntoClientArea en lui passant le handle de la fenêtre et l'objet MARGINS.

Voici un exemple de code afin d'étendre la marge gauche à toute la zone cliente (c'est-à-dire étendre la transparence à toute la fenêtre):

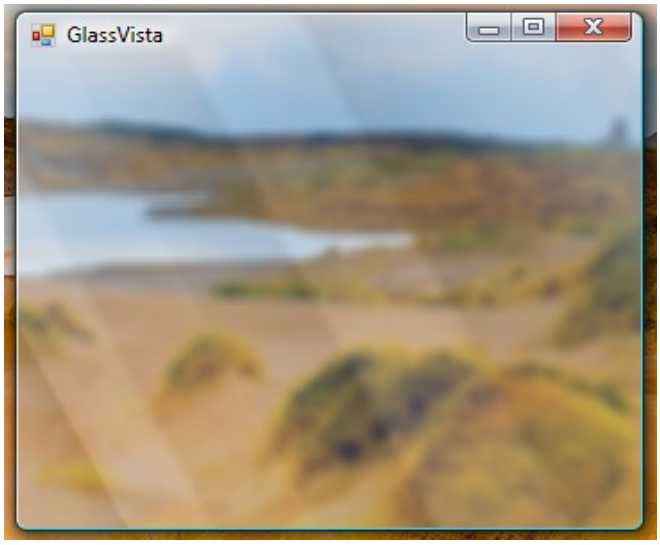
```
if (WrappersDWM.DwmIsCompositionEnabled()){
    //on peut activer la transparence
    glassMarges = new WrappersDWM.MARGINS(-1, 0, 0, 0);

    WrappersDWM.DwmExtendFrameIntoClientArea(this.Handl
e, ref glassMarges);
    this.Invalidate();  //pour forcer le repaint
}
```

Notez l'appel à la méthode Invalidate qui permet de repeindre la fenêtre. En effet, la dernière chose à faire est de peindre les zones que l'on souhaite voir transparentes avec un brush noir. Pour cela on redéfinit la méthode OnPaint de la fenêtre.

Le code ci-dessous permet de peindre l'ensemble de la zone cliente en noir:

```
protected override void OnPaint(PaintEventArgs e){
    if (WrappersDWM.DwmIsCompositionEnabled()){
        e.Graphics.FillRectangle(Brushes.Black,
this.ClientRectangle);
    }


    base.OnPaint(e);
}
```

Tout cela nous permet d'obtenir le résultat suivant:



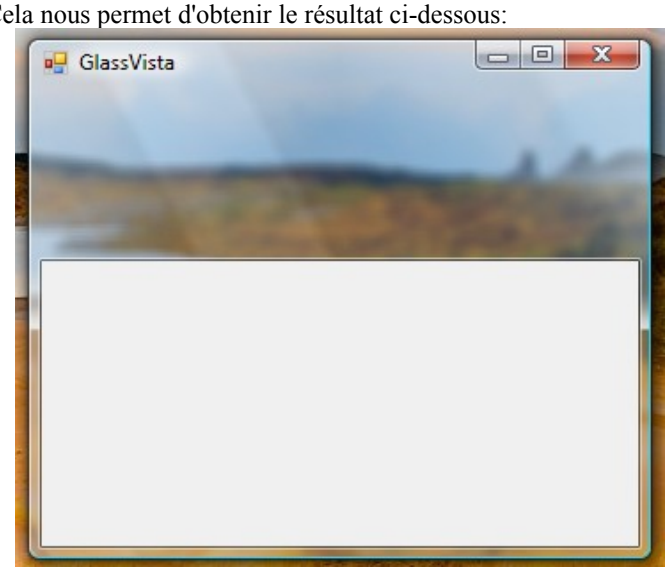
Prenons un deuxième exemple où l'on ne souhaite rendre transparent qu'un bandeau de 100 pixels de hauteur en haut de la fenêtre.
Il nous faut construire l'objet MARGINS de cette façon:

```
if (WrappersDWM.DwmIsCompositionEnabled()){
    //on peut activer la transparence
    glassMarges = new WrappersDWM.MARGINS(0, 0, 100,
0);

    WrappersDWM.DwmExtendFrameIntoClientArea(this.Handl
e, ref glassMarges);
    this.Invalidate();  //pour forcer le repaint
}
```

Puis modifier la fonction OnPaint pour ne peindre en noir que ce bandeau:

```
protected override void OnPaint(PaintEventArgs e){
    if (WrappersDWM.DwmIsCompositionEnabled()){
        e.Graphics.FillRectangle(Brushes.Black,
Rectangle.FromLTRB(0, 0,
            this.ClientRectangle.Width,
glassMarges.cyTopHeight));
    }

    base.OnPaint(e);
}
```

Cela nous permet d'obtenir le résultat ci-dessous:



6.2. Deuxième technique

La plupart des personnes qui souhaitent ajouter de la transparence à leur fenêtre vont probablement se satisfaire de la première méthode. Il existe néanmoins une autre méthode permettant d'avoir plus de contrôle sur la façon dont l'effet de transparence est construit. Il ne s'agit pas ici d'utiliser les bordures de la fenêtre pour les étendre mais d'indiquer une région de la fenêtre à rendre floue.

Nous allons utiliser ici la fonction DwmEnableBlurBehindWindow dont voici la signature non gérée:

```
HRESULT DwmEnableBlurBehindWindow(
    HWND hWnd,     const DWM_BLURBEHIND *pBlurBehind );
```

Et son wrapper:

```
[DllImport("dwmapi.dll", PreserveSig = false)]
public static extern void
DwmEnableBlurBehindWindow(IntPtr hWnd, ref
DWM_BLURBEHIND pBlurBehind);
```

Cette fonction ressemble à la fonction DwmExtendFrameIntoClientArea sauf que la structure à passer en paramètre est différente. En voici d'ailleurs la signature en code non managé:

```
typedef struct _DWM_BLURBEHIND {
DWORD dwFlags;
BOOL fEnable;
HRGN hRgnBlur;
BOOL fTransitionOnMaximized;
} DWM_BLURBEHIND, *PDWM_BLURBEHIND;
```

Et son équivalent en code managé:

```
[StructLayout(LayoutKind.Sequential)]
public struct DWM_BLURBEHIND{
    public uint dwFlags;
    [MarshalAs(UnmanagedType.Bool)]
    public bool fEnable;
    public IntPtr hRegionBlur;
    [MarshalAs(UnmanagedType.Bool)]
    public bool fTransitionOnMaximized;

    public const uint DWM_BB_ENABLE = 0x00000001;
    public const uint DWM_BB_BLURREGION = 0x00000002;
    public const uint DWM_BB_TRANSITIONONMAXIMIZED =
0x00000004;
}
```

Nous allons voir un peu plus loin à quoi vont servir les constantes déclarées.

Voici un descriptif des différents champs:

- **fEnable** : booléen à mettre à vrai si l'on veut appliquer la transparence.
- **hRegionBlur** : représente la région à rendre floue.
- **fTransitionOnMaximized** : indique si l'effet de transparence doit devenir opaque quand une fenêtre (n'importe laquelle) du bureau est maximisée. Pour mieux comprendre, pensez à la barre des tâches de Vista: en temps normal elle est transparente, mais si on maximise une fenêtre elle devient opaque.
- **dwFlags** : ce champ est une combinaison de constantes permettant d'indiquer quels membres (parmi les trois précédents) ont été définis. Les constantes à utiliser sont celles déclarées dans le code de la structure DWM_BLURBEHIND plus haut. Ainsi, si l'on renseigne les trois champs ci-dessus, il faudra remplir ce membre de cette façon:

```
dwFlags = WrappersDWM.DWM_BLURBEHIND.DWM_BB_ENABLE |
          WrappersDWM.DWM_BLURBEHIND.DWM_BB_BLURREGION
|
          WrappersDWM.DWM_BLURBEHIND.DWM_BB_TRANSITIONO
NMAXIMIZED;
```

Pour construire une région (objet de type Region) on passe généralement par un objet GraphicsPath. N'oubliez pas d'importer l'espace de nom System.Drawing.Drawing2D.

Voici un exemple de code où l'on construire une région de forme ovale:

```
GraphicsPath gp = new GraphicsPath();
gp.AddEllipse(this.Width / 2 - 150, 30, 300, 150);
Region regionBlur = new Region(gp);
```

Pour illustrer cette technique nous allons construire une fenêtre un peu spéciale. Elle ne sera pas rectangulaire comme les fenêtres par défaut, mais ovale et sans bordure ni barre de titre. Bien sûr on lui appliquera l'effet de transparence.

Pour mémoire, définir une forme particulière à une fenêtre s'effectue en lui assignant une nouvelle valeur à sa propriété Region (de type Region).

Ainsi, rendre une fenêtre de forme elliptique se fera de cette manière:

```
GraphicsPath gp = new GraphicsPath();
gp.AddEllipse(this.Width / 2 - 150, 30, 300, 150);
Region regionBlur = new Region(gp);
this.Region = regionBlur;
```

Mettons maintenant tout ceci en pratique. Voici le code permettant de créer une fenêtre ovale et transparente:

```
//on vérifie si la composition est activée ou non
if (WrappersDWM.DwmIsCompositionEnabled()){
    //on peut activer la transparence
    using (Graphics gc = CreateGraphics()){
        GraphicsPath gp = new GraphicsPath();
        gp.AddEllipse(this.Width / 2 - 150, 30, 300,
150);
        Region regionBlur = new Region(gp);

         //on rend la fenêtre ovale en utilisant la
même Region
        this.Region = regionBlur;

        WrappersDWM.DWM_BLURBEHIND bbh = new
WrappersDWM.DWM_BLURBEHIND();
        bbh.dwFlags =
WrappersDWM.DWM_BLURBEHIND.DWM_BB_ENABLE |
                    WrappersDWM.DWM_BLURBEHIND.DWM_BB
_BLURREGION |

WrappersDWM.DWM_BLURBEHIND.DWM_BB_TRANSITIONONMAXIMIZED
;

        bbh.fEnable = true;
        bbh.hRegionBlur = regionBlur.GetHrgn(gc);
        bbh.fTransitionOnMaximized = false;
        WrappersDWM.DwmEnableBlurBehindWindow(this.Hand
le, ref bbh);
```

```
    }
    this.Invalidate();  //pour forcer le repaint
} else {
    MessageBox.Show("Composition non activée");
    //pas de transparence ici !
}
```

Bien sûr, il vous faut toujours peindre le fond de la fenêtre en noir dans la méthode OnPaint:

```
protected override void OnPaint(PaintEventArgs e){
    if (WrappersDWM.DwmIsCompositionEnabled()){
        e.Graphics.Clear(Color.Black);
    }
    base.OnPaint(e);
}
```

Au final nous obtenons le résultat ci-dessous:

7. Dessiner sur la transparence

Maintenant que votre fenêtre est transparente, vous voudriez sans doute la garnir et y ajouter du texte.

Vous allez malheureusement être confronté à quelques petits soucis:

Voici un label simple avec du texte en noir à l'intérieur.

Le même label mais avec le fond transparent. Pas génial.

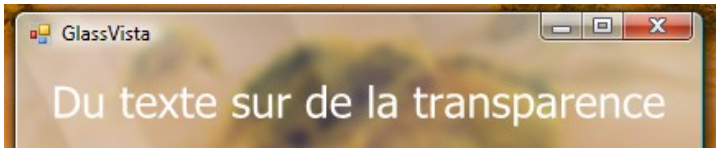
Encore le même mais avec un fond noir (c'est la couleur qui détermine les zones transparentes). Ce n'est pas encore ça.

Une solution ici est de ne pas utiliser de label mais de peindre directement le texte que l'on souhaite afficher en utilisant les classes du GDI+. Par contre n'utilisez pas la méthode DrawString de la classe Graphics. Il faut passer par l'intermédiaire d'un objet GraphiquePath.

Voici un exemple de code que vous pouvez mettre dans méthode OnPaint:

```
Graphics g = this.CreateGraphics();
GraphicsPath blackfont = new GraphicsPath();
SolidBrush brsh = new SolidBrush(Color.White);

blackfont.AddString("Du texte sur de la transparence",
    new FontFamily("Tahoma"), (int)FontStyle.Regular,
26,
    new Point(10, 10), StringFormat.GenericDefault);

g.SmoothingMode =
System.Drawing.Drawing2D.SmoothingMode.HighQuality ;
g.FillPath(brsh, blackfont);
```

Et le résultat obtenu:



Il faut utiliser la même technique pour les images (pictureBox déconseillée). Vous pouvez par contre ici simplement utiliser la fonction DrawImage de la classe Graphics.

Une simple ligne ajoutée à la méthode OnPaint suffit pour afficher une image qui se trouve en ressource:

```
protected override void OnPaint(PaintEventArgs e){
    if (WrappersDWM.DwmIsCompositionEnabled()){
        e.Graphics.FillRectangle(Brushes.Black,
this.ClientRectangle);
        //affichage de l'image
        e.Graphics.DrawImage(global::GlassVista.Prope
rties.Resources.logo, new Point(10, 60));
    }
    base.OnPaint(e);
 }
```

Et le résultat obtenu:

8. Conclusion

Notre petit tour dans le monde de la transparence touche à sa fin. Nous avons vu les bases pour la création de fenêtres WinForm avec l'effet Aero Glass de Windows Vista. N'hésitez pas à consulter les différents sites Web référencés ci-dessous pour plus d'informations.

Retrouvez l'article de Vicent Lainé : Lien25

Les derniers tutoriels et articles

Les conversions numériques en C,C++,C++/CLI

Les conversions numériques sont des sujets qui reviennent souvent sur le forum Visual C++/MFC Il faut dire que les conversions de chaînes de caractères vers un type natif et vice-versa sont omniprésentes dans nos développements Windows. A travers cet article je vous propose de faire le point sur les techniques disponibles pour ces travaux.

1. Définition du problème

Comme je l'ai dit en introduction la conversion entre chaine de caractères et type natif est centrale dans nos programmes,
Les contrôles Windows travaillent avec des chaines de caractères tout en pouvant représenter une saisie numérique voir décimale.
La représentation interne de ces données visuelles étant le plus souvent stockée en interne dans leur type natif,
Vient alors à se poser le problème de la conversion entre le contrôle et sa variable.

La majorité des posts s'y rapportant sur nos forums se résume souvent à ces questions:
Comment faire pour convertir une chaine en entier, double , float etc..
Comment formater une chaine à partir d'un entier, double, float etc..
Comment récupérer un entier, double float, etc., à partir d'un contrôle, et bien sûr l'inverse qui consistera à mettre à jour le contrôle à partir d'un type natif.

Pour procéder nous disposons de plusieurs techniques issues de différentes bibliothèques :
Celles du C , les bibliothèques standards du C++, mais aussi quelques apports intéressants avec le projet BOOST.
j'ai aussi fait attention à ce que l'ensemble des codes présentés fonctionnent en **UNICODE**.
Enfin je ne pouvais ignorer le C++/CLI, je consacrerai donc une rubrique pour les deux sens de conversion.
C'est ce que nous allons découvrir maintenant.

2. Transformation d'une CString ou du contenu d'un contrôle vers un type natif

2.1. La Bibliothèque C

La bibliothèque C fournit un ensemble de fonctions permettant la transformation de chaine en type de données et inversement.
Un post général y est consacré dans la faq Visual C++, aussi je ne vais pas m'attarder dessus, je préfère donner la priorité au traitement C++ du sujet :
Comment convertir une CString en int, double, long ? : Lien26

2.2. Du coté des MFC

Les MFC comme l'api 32 permettent avec GetDlgItemInt (Lien27) la récupération du contenu d'un contrôle sous forme d'entier.
mais quid des autres types long,float,double ?
Pour ces autres types il faudrait attacher au contrôle une variable correspondante au type de donnée souhaité et appeler la méthode UpdateData(TRUE) (Lien28) pour disposer de sa valeur.

Voyons maintenant une méthode de transformation d'une chaine dans son type natif mettant en œuvre la bibliothèque standard C++ (standard library).

2.3. La bibliothèque standard du C++ (SL)

Commençons par convertir une chaine vers un type spécifié :

```
#include <string>
#include <iostream>
#include <sstream>

template<typename T,typename S>
bool FromString( const S & Str, T & Dest ){
#ifdef _UNICODE
    std::wistringstream iss( Str );
#else
        std::istringstream iss( Str );
#endif
    // tenter la conversion vers Dest
    return iss >> Dest != 0;
}
```

Utilisation:

```
int nInt ;
FromString(_T("10"), nInt ); // conversion en int.

double dDouble;
FromString( _T("3.14107"), dDouble ); // conversion en
double

CString str=_T("1200");
#ifdef _UNICODE
        std::wstring strstl;
#else
        std::string strstl;
#endif

strstl=_T("1200");
int n=0;
FromString(str.GetString(),n);
n=0;
FromString(strstl,n);
```

La version associée à un contrôle MFC :

La récupération du contenu d'un contrôle dans son type natif : entier, double, float, long ne cause pas plus de problèmes en modifiant un peu le code précédent on obtient :

```
#include <sstream>
#include <string>
#include <iostream>

template<typename T>
bool FromCtrl( const CWnd & Ctrl, T & Dest ){
    CString Str;
    Ctrl.GetWindowText(Str);
#ifdef _UNICODE
    std::wistringstream iss( static_cast<LPCTSTR>( Str)
);
#else
       std::istringstream iss( static_cast<LPCTSTR>(
Str) );
#endif
    // tenter la conversion vers Dest
    return iss >> Dest != 0;
}

//Utilisation:
double d;
FromCtrl(*GetDlgItem(IDC_EDITNOM),d);
if(d==10.0){
// traitement
}
// etc...
```

Ces exemples s'appuient sur la classe istringstream et utilisent l'opérateur surchargé >> pour effectuer la conversion.
Ce type de fonction se nomme fonction modèle, elle permet de rester générique par rapport au type T utilisé.

2.4. Le Projet BOOST

Voyons maintenant ce que propose le projet BOOST :
Il fournit une classe de conversion lexical_cast qui permet la conversion de chaines représentant des nombres en base 10.

Pour utiliser l'exemple qui va suivre, vous devrez télécharger le projet BOOST (Lien29) disponible sur SourceForge La dernière version stable est la 1.33.1
Vous pouvez aussi utiliser la version graphique de distribution de BOOST avec un outil distribué par Boost Consulting (Lien30) ,voir aussi le tutoriel Installer et utiliser Boost sous Windows avec Visual C++ 2005 (Lien31)
Commençons par convertir une chaine vers un type spécifié :
L'utilisation de lexical_cast est confondante de simplicité :

```
#include <string>
#include <boost/lexical_cast.hpp>
int n,n1,n2;
CString str=_T("1200");
#ifdef _UNICODE
       std::wstring strstl;
#else
       std::string strstl;
#endif
       strstl=_T("1200");
    try{
               n =
boost::lexical_cast<int>(static_cast<LPCTSTR>(str));
               n1 = boost::lexical_cast<int>(strstl);
               n2 =
boost::lexical_cast<int>(_T("1200"));
       }
       catch(boost::bad_lexical_cast &e)
       {
#ifdef _DEBUG
               TRACE("Mauvaise conversion :
%s",static_cast<const char *>(e.what()));
#endif
               return false;
       }
```

Plus besoin de flux, la séquence peut être utilisée directement.
Néanmoins on peut toujours encapsuler le traitement dans une fonction modèle :

```
#include <string>
#include <boost/lexical_cast.hpp>
template<typename T,typename S>
bool BoostFromString(const S &rStr, T & Dest ){
    try{
               Dest = boost::lexical_cast<T>(rStr);
       }
       catch(boost::bad_lexical_cast &e)
       {
#ifdef _DEBUG
               TRACE("Mauvaise conversion :
%s",static_cast<const char *>(e.what()));
#endif
               return false;
       }
       return true;
}
```

Le même exemple donnera :

```
int n=0;
// MFC
CString str=_T("1200");
BoostFromString(str.GetString(),n);
// C++ - STL : string
#ifdef _UNICODE
       std::wstring strstl;
#else
       std::string strstl;
#endif
strstl=_T("1200");
BoostFromString(strstl,n);
// une chaine
BoostFromString(_T("1200"),n);
```

La version associée à un contrôle MFC :

```
template<typename T>
bool BoostFromCtrl( const CWnd & Ctrl, T & Dest )
{
    CString Str;
    Ctrl.GetWindowText(Str);
    try
       {
               Dest =
boost::lexical_cast<T>(static_cast<LPCTSTR>(Str));
       }
       catch(boost::bad_lexical_cast &e)
       {
#ifdef _DEBUG
               TRACE("Mauvaise conversion :
%s",static_cast<const char *>(e.what()));
#endif
               return false;
       }
       return true;
}
```

Utilisation:

```
int n=0;
BoostFromCtrl(*GetDlgItem(IDC_EDITNUM),n);
```

A travers ces exemples nous avons découvert différentes techniques de conversion d'une chaine vers un type numérique, le contrôle de la conversion, et ce avec les différentes bibliothèques.
Pour ma part je trouve l'utilisation de **boost ::lexical_cast** plus séduisante car très simple et utilisable directement.

2.5. Le C++/CLI

En C++/CLI la conversion d'une chaine managée (String) en entier ou double est relativement simple :

```
#include "stdafx.h"

using namespace System;

int main(array<System::String ^> ^args){
    String ^str=L"1200,20";
    double d;
    int n;
    try
    {
        Console::Write(L"Tentative conversion double
");
        d=Convert::ToDouble(str);
        Console::Write(L"d:");
        Console::WriteLine(d);

        Console::Write(L"Tentative conversion Entier
");
        n=Convert::ToInt32(str);// provoque une
erreur !!!.
        Console::Write(L"n:");
        Console::WriteLine(n);
    }
    catch(FormatException ^e){
        Console::Write(e->Message);
    }
    return 0;
}
```

On trouvera les fonctions de conversions dans l'espace de nom System::Convert : Lien32
La conversion en double ou en int ne cause pas de problème particulier : Lien33
En cas de chaine invalide une exception FormatException est levée, comme c'est le cas ici avec mon exemple lors de la tentative de conversion de la chaine en entier.
Autre exemple mettant en oeuvre les conversions entre base:

```
String ^bin = L"1111";
int decimal = Convert::ToInt32(bin,2);
String ^hexa = Convert::ToString(decimal, 16);
Console::WriteLine(L"binaire = {0}\ndécimal =
{1}\nhexadécimail = {2}", bin, decimal, hexa);
```

3. Conversion d'un type int,long,float,double vers une chaine de caractères

Passons maintenant à l'exercice inverse qui consiste à prendre une valeur d'un type natif et de la transformer en chaine de caractères destinée à mettre à jour un contrôle.

3.1. La Bibliothèque C

Classiquement on pourra utiliser la fonction sprintf , ou les fonctions associées au type de données : comme itoa pour la conversion d'un int vers une chaine .
Une fois la chaine constituée on mettra à jour le contrôle.

3.2. Du Coté des MFC

La classe **CString** nous aide dans la conversion en proposant une méthode Format fonctionnant de la même manière que la fonction vsprintf, sprintf du C. Cet exemple de la faq Visual montre comment l'utiliser : Comment convertir un entier, un double, un float, etc, en chaîne de caractères ? (Lien34)
Toujours dans l'optique de mettre à jour directement le contrôle,
Pour les autres types il faudrait attacher au contrôle une variable correspondante au type natif souhaité et appeler la méthode UpdateData(TRUE) (Lien28) pour disposer de sa valeur.

3.3. La bibliothèque standard du C++ (SL)

Comme précédemment, le code qui suit permet de se passer de l'association d'une variable à un contrôle et donc d'affecter directement une valeur d'un type natif au contrôle désigné.

```
#include <sstream>
#include <string>
#include <iomanip>
#include <iostream>

class FormatNum{
public :
        FormatNum(){}
        template <typename T>
        FormatNum(const T&t){
                operator <<(t);
        }
        template <typename T>
    FormatNum & operator << (const T& t){
        m_ss << t;
        return *this;
    }

public :
#ifdef _UNICODE
        std::wstringstream m_ss;
#else
        std::stringstream m_ss;
#endif
};

template<typename T>
void ToCtrl(CWnd & Ctrl,const T & Src,FormatNum
&rFormat=FormatNum()){
        rFormat << Src;

#ifdef _UNICODE
        std::wstring s=rFormat.m_ss.str();
#else
        std::string s=rFormat.m_ss.str();
#endif
    Ctrl.SetWindowText(s.c_str());
}
```

Utilisation :

```
// met 10.345 dans le contrôle.
ToCtrl(*GetDlgItem(IDC_EDITNUM),10.345);
// met 10.35 dans le contrôle.
ToCtrl(*GetDlgItem(IDC_EDITNUM),10.345,

        FormatNum()<<std::setprecision(4));
```

Le traitement se décompose en deux parties :
La fonction modèle ToCtrl permettant la transformation du type utilisateur et l'affectation de la chaine au contrôle passée en argument.
Un objet fonction FormatNum optionnel qui permet le formatage du flux pour contrôler la conversion, l'enchainement des arguments, et qui fournit l'objet flux de conversion de la classe

stringstream à la fonction modèle ToCtrl.

Voyons son utilisation dans les exemples qui suivent :
Dans le cas d'un double ou float si on veut maitriser la précision du nombre envoyé on pourra utiliser la fonction setprecision définie dans l'entête standard **iomanip** pour fixer le nombre de digits souhaités.
Vous pouvez bien-sûr utiliser les autres fonctions et compléter le flux
Exemples:
Contrôler la longueur de la chaine créée, et spécifier un caractère de remplissage.

```
// donne 0000010.35
ToCtrl(*GetDlgItem(IDC_EDITNUM),
          10.345,

FormatNum()<<std::setprecision(4)<<std::setfill(_TCHAR(
'0'))<<std::setw(10));
```

Cet exemple impose une précision de 4 digits, une chaine de 10 caractères remplie avec des '0'.
Dans le même ordre d'idée on pourra fixer la base de conversion ...

```
// donne 0000000020
ToCtrl(*GetDlgItem(IDC_EDITNUM),
       32,

FormatNum()<<std::setprecision(4)<<std::setfill(_TCHAR(
'0'))<<std::setw(10)<<setbase(16));
```

Enfin rajouter du texte devant la conversion :

```
// donne: Conversion Hexa: 0x0000000020
ToCtrl(*GetDlgItem(IDC_EDITNUM),
32,
FormatNum()<<_T("Conversion Hexa: 0x")
<<std::setprecision(4)<<std::setfill(_TCHAR('0'))<<std:
:setw(10)<<setbase(16));
//ou
ToCtrl(*GetDlgItem(IDC_EDITNUM),
32,
FormatNum(_T("Conversion Hexa:
0x"))<<std::setprecision(4)<<std::setfill(_TCHAR('0'))<
<std::setw(10)<<setbase(16));
```

Ou encore une syntaxe plus aérée:

```
FormatNum format;
format <<_T("Conversion Hexa: 0x")
<<std::setprecision(4)<<std::setfill(_TCHAR('0'))<<std:
:setw(10)<<setbase(16);

ToCtrl(*GetDlgItem(IDC_EDITNUM),32,format); // donne:
Conversion Hexa: 0x0000000020
```

Vous noterez aussi l'utilisation optionnelle de la spécification du format de conversion (FormatNum).
On pourra compléter notre traitement par une fonction de conversion vers une CString ou string de la STL.

```
template<typename T,typename S>
void ToString(S & rstr,const T & Src,FormatNum
&rFormat=FormatNum())
{
       rFormat << Src;
#ifdef _UNICODE
       std::wstring s=rFormat.m_ss.str();
#else
       std::string s=rFormat.m_ss.str();
#endif
       rstr=s.c_str();
}
```

Utilisation :

```
CString str;
ToString(str,1200);
#ifdef _UNICODE
       std::wstring strstl;
#else
       std::string strstl;
#endif
ToString(strstl,1200);
```

3.4. Le Projet BOOST

BOOST fournit la bibliothèque **Boost Format** contenant une classe format permettant de réaliser les conversions, de plus elle autorise les spécifications de formats comme printf du C.

```
try{
#ifdef _UNICODE
       boost::wformat f(_T("c'est %1% maniere de %2%
les %3%"));
#else
       boost::format f("c'est %1% maniere de %2% les
%3%");
#endif
       f % 1;
       f % _T("voir") % _T("choses");
       AfxMessageBox(boost::io::str(f).c_str());

       f.clear(); // vider le tampon avant
reutilisation...

       f.parse(_T("conversion entiere %d"));
       f % 1200;

       AfxMessageBox(boost::io::str(f).c_str());

       #ifdef _UNICODE
       GetDlgItem(IDC_EDITNUM)-
>SetWindowText(boost::io::str(boost::wformat(_T("Conver
sion Hexa: %010x")) % 32).c_str());
       #else
       GetDlgItem(IDC_EDITNUM)-
>SetWindowText(boost::io::str(boost::format("Conversion
Hexa: %010x") % 32).c_str());
       #endif
}
catch(boost::io::format_error &e)
{
TRACE("Mauvaise conversion : %s",static_cast<const char
*>(e.what()));
}
```

Boost Format peut être utilisée de plusieurs manières (exemples ci-dessus).
- En remplacement d'arguments: chaque nombre entouré de % doit être remplacé en utilisant l'opérateur % .
- En spécifiant un format: On utilisera alors les mêmes spécifications de format disponibles avec la fonction printf du C.
Je n'ai montré que deux aspects de cette bibliothèque qui possède de nombreuses possibilités.

L'intérêt de **Boost Format** dans un projet **MFC** peut sembler plus limité puisque nous disposons de la méthode Format de la classe CString.
Elle sera beaucoup plus utile dans des projets C++ standards, ou dans la réalisation de bibliothèques devant se passer des MFC.

3.5. Le C++/CLI

La conversion d'un type natif vers une string se fait très naturellement en C++/CLI :

```
#include "stdafx.h"
using namespace System;

int main(array<System::String ^> ^args){
    double d=3.14107;
    int    n=100;
    String ^Str;
    Console::Write(L"Conversion double en chaine :");
    Str=d.ToString();
    Console::WriteLine(Str);

    Console::Write(L"Conversion int en chaine :");
    Str=n.ToString();
    Console::WriteLine(Str);

    Console ::WriteLine((12356).ToString()) ;
    Console ::WriteLine((0xFF).ToString());
    Console ::WriteLine((true).ToString());
}
```

d.ToString() mérite quelques explications :
En C++/CLI tous les types classiques sont des alias d'objet du framework .net, un int est donc un objet qui hérite de la classe System::Objet.
Celle-ci possède (entre autre) la méthode ToString() qui permet de convertir l'objet en chaine de caractère.
Autre avantage en C++/CLI les types classiques étant des objets ils sont initialisés à zéro.

Une autre possibilité intéressante qui est démontrée dans les dernières lignes de mon exemple :
Un littéral numérique peut être considéré comme un objet s'il est entouré de parenthèses, alors la méthode ToString peut être appliquée…
Cette forme de conversion a l'avantage d'être simple mais elle ne permet de contrôler le format de la chaine obtenue .
On utilisera alors String::Format.

```
    using namespace System;
    using namespace System::Globalization;

    Str=String::Format(CultureInfo::CurrentCulture,
        "(C) Currency: . . . . . {0:C}\n" +
        "(D) Decimal:. . . . . {0,4:0}\n" +
        "(E) Scientific: . . . . . {1:E}\n" +
        "(F) Fixed point:. . . . . {1:F}\n" +
        "(G) General:. . . . . {0:G}\n" +
        "    (default):. . . . . {0} (default =
'G')\n" +
        "(N) Number: . . . . . {0:N}\n" +
        "(P) Percent:. . . . . {1:P}\n" +
        "(R) Round-trip: . . . . . {1:R}\n" +
        "(X) Hexadecimal:. . . . . {0:X}\n",
        100, 3.14107);
    Console::WriteLine(Str);

    Str=String::Format( "3.14107->: {0:#,#.##;}",
3.14107);//3.14
    Console::WriteLine(Str);
    Str=String::Format( "3.555->: {0:#,#.##;}",
3.555);//3.56
    Console::WriteLine(Str);
    Str=String::Format( "100->: {0,5:00###}", 100 );
//00100
    Console::WriteLine(Str);
```

Le format se décompose de la manière suivante : {index[,alignment][:formatString]}
Vous retrouverez l'ensemble des descripteurs de format (Lien35) dans la documentation MSDN.

4. Conclusion

Il ne vous reste plus qu'à adopter la solution qui convient le mieux à vos besoins ou à la situation.

Retrouvez l'article de farscape en ligne : Lien36

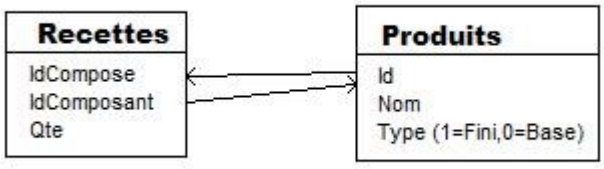
Les derniers tutoriels et articles

DB2 et le SQL récusif.

Cet article a pour vocation d'expliquer le fonctionnement du SQL récursif. Ce tutoriel est basé sur DB2 mais il est globalement applicable à d'autres gestionnaires de bases de données comme par exemple SQL Server. Cette technologie est à la fois simple et très puissante et permet d'éviter des procédures stockées complexes ou d'implémenter la récursivité au niveau du client qui entraine un important trafic de données sur le réseau.

1. Présentation de l'exemple

Pour illustrer les possibilités du SQL récursif dans DB2, prenons l'exemple classique de la composition d'un produit fini. Le produit fini est composé de produits de base mais également de produits semi-finis. Un produit fini peut également être un produit semi-fini pour un autre produit.



L'approche la plus simple pour obtenir des informations comme "Quels produits de base dois-je prendre pour réaliser complètement mon produit ?" ou "Quelle est la composition d'un produit et de chaque produit qui le compose ?" est évidemment la récursivité. Vous pouvez bien sûr la développer dans le programme client. Ce qui entrainera un important trafic entre le client et le serveur. Une possibilité souvent méconnue ou oubliée est l'utilisation du SQL récursif.

2. La syntaxe

L'instruction permettant de réaliser un "Select" récursif est divisé en quatre parties:

- La première partie définit la table temporaire qui recevra le résultat tout d'abord du "Select" d'initialisation et ensuite des résultats successifs du "Select" de récursion.
- La seconde est le "Select" d'initialisation.
- La troisième est le "Select" qui sera exécuté de manière récursive.
- La quatrième partie est le "Select" final qui est exécuté sur la table temporaire

La syntaxe du SQL récursif

```
WITH
        nomDeLaTableTemporaire (champ1, champ2, champ3,
...) AS
        (
                SELECT d'initialisation

                UNION ALL

                SELECT de récursion
        )
        SELECT de résultat
```

Concrètement, c'est comme si la table temporaire recevait le résultat du premier select. Ensuite, pour chaque ligne contenue dans cette table temporaire, le "Select" de récursion est exécuté. Le résultat est ajouté à la table temporaire et servira donc d'entrée pour autant de nouvelle exécution du "Select" que de lignes ajoutées et ainsi de suite. Au niveau moteur DB2, il est probable que cela ne se passe pas exactement de cette manière mais c'est l'idée générale.

3. Exemple: La liste des produits de base

Pour obtenir la liste des produits de base nécessaires à la réalisation du produit fini dont l'Id est 1, la commande SQL sera:

Commande SQL

```
WITH
   ProduitsDeBase (Id, Qte) AS
   (
       SELECT Racine.IdComposant, Racine.Qte
           FROM Recettes Racine
           WHERE Racine.IdCompose = 1

       UNION ALL

       SELECT Enfant.IdComposant, Enfant.Qte *
Parent.Qte
           FROM ProduitsDeBase Parent, Recettes Enfant
           WHERE Parent.Id = Enfant.IdCompose
   )
   SELECT ProduitsDeBase.Id, Sum(Qte), Produits.Nom
       FROM ProduitsDeBase, Produits
       WHERE ProduitsDeBase.Id = Produits.ID AND
Produits.Type = 0
       GROUP BY ProduitsDeBase.Id, Produits.Nom
       ORDER BY Produits.nom
```

Dans le "Select" de récursion, une jointure est réalisée sur la table parent (table temporaire) de manière à obtenir tout les composants d'un élément. Le type de produit sera nécessaire pour la sélection finale.

Notez que pour obtenir le nombre total d'élément, il est nécessaire de multiplier la quantité du composant par la quantité du composé.

Dans notre exemple, le produit fini 'A' concerné est composé de 2 produits 'B', 1 'C' et 1 'D'. Le produit 'C' est lui même composé de 2 'D' et 3 'E'. 'B' quant a lui est composé de 1 'E', 2 'F' et 2 'C'. Le

résultât de la requête donne donc:

Résultat

```
ID          2           NOM
----------- ----------- ------------------------------
SQL0347W  The recursive common table expression
"DB2ADMIN.PRODUITSDEBASE" may
contain an infinite loop.  SQLSTATE=01605

          4           6 D
          5          17 E
          6           4 F

  3 record(s) selected with 1 warning messages printed.
```

Notez le message d'avertissement. Nous verrons dans le chapitre suivant pourquoi ce message est affiché et comment éviter ce problème.

4. La récursivité maitrisée

Imaginons que dans la recette, un produit fini contienne un autre produit fini qui lui même contient le premier. Soit A composé de B et B composé de A. Dans ce cas, la récursivité va entrer en boucle.

Pour éviter ce piège, vous devez limiter la profondeur de la récursivité en utilisant un compteur. Dans l'exemple, nous allons utiliser une colonne appelée "Niveau". Elle est initialisée à 0 et ensuite incrémentée de 1 à chaque appel récursif. Une condition sur cette colonne permet alors de bloquer la récursivité. A vous de choisir un nombre suffisamment grand pour que le traitement puisse être complet mais suffisamment petit pour éviter une boucle très pénalisante en temps en cas d'anomalie dans les données.

Commande SQL

```
WITH
   ProduitsDeBase (Niveau,Id, Qte) AS
   (
       SELECT 0, Racine.IdComposant, Racine.Qte
           FROM Recettes Racine
           WHERE Racine.IdCompose = 1

       UNION ALL

       SELECT Parent.Niveau + 1, Enfant.IdComposant,
Enfant.Qte * Parent.Qte
           FROM ProduitsDeBase Parent, Recettes Enfant
           WHERE Parent.Id = Enfant.IdCompose AND
Parent.Niveau < 10
   )
   SELECT ProduitsDeBase.Id, Sum(Qte), Produits.Nom
       FROM ProduitsDeBase, Produits
       WHERE ProduitsDeBase.Id = Produits.ID AND
Produits.Type = 0
       GROUP BY ProduitsDeBase.Id, Produits.Nom
       ORDER BY Produits.nom
```

Résultat

```
ID          2           NOM
----------- ----------- ------------------------------
          4           6 D
          5          17 E
          6           4 F

  3 record(s) selected.
```

La profondeur maximum a, dans ce cas, été limitée à dix.

5. Conclusion

Souvent méconnu, le SQL récursif est un outil puissant et pratique qui permet d'économiser un trafic important entre le serveur et le client qui autrement aurait du assurer la récursivité par une succession de lecture. Toutefois, il est impératif de contrôler la récursion sous peine de voir le serveur entrer en boucle ou créer une table intermédiaire d'une taille astronomique.

Retrouvez l'article de Jean-Alain Baeyens en ligne : Lien37

Sécurisez votre instance Oracle

...ou comment compliquer la vie des pirates... sans se laisser envahir par le côté obscur de la force.

1. Introduction

Que ce soit dans les anciennes ou les nouvelles implémentations, le besoin de revoir les aspects sécuritaires de ces environnements s'avère nécessaire, tant du point de vue de la confidentialité d'accès aux données que de l'intégrité de celles-ci.

Les lois européennes sur la protection des données sont claires sur le fait qu'il incombe de veiller à la protection des données personnelles que l'on stocke.

D'autres informations sont jugées sensibles au sein des entreprises, comme par exemple la liste des clients, le prix des produits ou les salaires des collaborateurs de l'entreprise.

La menace interne est généralement sous-estimée et donc souvent peu prise en compte, alors que la majorité des analyses disponibles sur le sujet indiquent que la sécurisation périmétrique ne suffit plus, une grande partie des menaces provenant désormais de l'interne.

Le but de cet article est la revue des menaces impliquant les informations hébergées au sein des instances Oracle et de proposer un catalogue de mesures permettant d'y remédier ou tout le moins d'en minimiser les risques.

"Mieux vaut connaître son ennemi pour le combattre !" Conscient que donner ce type d'information sur des brèches sécuritaires peut pousser certaines personnes mal intentionnées à s'y engouffrer, je ne peux que vous conseiller de ne pas "plonger duc ôté obscur de la force"... et d'envisager ce type d'information de manière la plus professionnelle qu'il soit.

2. Sources

Le but principal est la diminution sensible des risques d'accès ou de modification inappropriée des données. Il se base principalement sur les sources suivantes:

- les compétences et expériences des DBA
- les différents sénari d'intrusions disponibles sur Internet
- les outils BackTrack 2 : Lien38
- les informations reçues lors du cours Oracle anti-hacking donné par M. Alexander Kornbrust, un des spécialistes de la sécurité sur Oracle : Lien39
- le site de Pete Finnigan : Lien40

- la documentation officielle d'Oracle : Oracle Database Security Guide : Lien41
- les diverses autres documentations délivrées par Oracle sur le sujet (via Metalink, Grid control, etc.)
- la Faq Oracle Sécurité de developpez.com : Lien42

3. Types de dangers

Les dangers peuvent être catégorisés de la manière suivante :

No	Type de danger	Explication
1	Déni de service	Plantage volontaire du SGBDR rendant l'accès aux données impossible
2	Intrusion	Accès malintentionné afin d'obtenir plus d'informations que nécessaire
3	Injection SQL	Utilisation de failles du SGBDR afin de s'octroyer des droits excessifs
4	Corruption de données	Pourrissement des données métier par mauvaise configuration, envie de nuire, ...

4. Dangers et mesures

Ici sont exposés les dangers latents et les mesures à apporter afin de les éviter.

4.1. Schémas / Logins

Oracle mixte la notion de schéma et de login. Un Schéma est en fait lié à un utilisateur qui est propriétaire d'au moins un objet. Un schéma, dans son rôle de conteneur d'objets devrait en fait être un login verrouillé dont le mot de passe ne devrait même pas être connu.

Pour les schémas métier créés par vos équipes de développement, c'est généralement le cas. Personne ne se connecte directement sur ces logins, mais un utilisateur spécifique et applicatif peut y accéder.

Dans les fait malheureusement, la plupart des applicatifs externes n'utilisent pas le prédicat du schéma pour accéder aux tables. Il leur faut donc soit se connecter directement sous le nom du schéma, soit utiliser la notion de synonyme. Couplé à un trigger sur connexion, la commande ALTER SESSION SET CURRENT_SCHEMA offre cependant une alternative élégante et permet la dissociation du schéma et du login utilisé.

Pour couronner le tout, certains logins/schémas système Oracle sont installés par défaut dans toute nouvelle instance et ont des droits excessifs ou/et des failles sécuritaires. Il est donc primordial de ne laisser sur une instance Oracle que les schémas nécessaires. Au besoin, il est toujours possible de recréer ces schémas via les scripts Oracle. Le surplus doit être désinstallé ou, au minimum, verrouillé.

La liste des schémas Oracle, une explication sommaire et les scripts de création et de désinstallations sont décrits dans l'article Les schémas Oracle (Lien43).

4.2. Mots de passe

4.2.1. Mots de passe simplistes

Type	2 (intrusion)
Contrôle	Vérification régulière à l'aide d'outils Backtrack (checkpwd + francais.txt), basé sur la comparaison de chaînes hachées. Vérification périodiques en brute force
Mesure	Demander aux utilisateurs de modifier les mots de passe faibles; forcer la complexification des mots de passe via les profiles Oracle

```
CREATE PROFILE Compexite_Pwd_profile
        LIMIT PASSWORD_LIFE_TIME 30
        LIMIT PASSWORD_REUSE_TIME 180
        LIMIT PASSWORD_REUSE_MAX 3 ;

ALTER USER scott PROFILE my_profile;
```

```
C:\oracle_checkpwd>checkpwd122.exe
system/monpwd@hostdvp:1521/ORA10G.DVP.COM
default_passwords.txt
Checkpwd 1.22 - (c) 2007 by Red-Database-Security GmbH
Oracle Security Consulting, Security Audits & Security
Trainings
http://www.red-database-security.com

initializing Oracle client library
connecting to the database
retrieving users and password hash values
disconnecting from the database
opening weak password list file
reading weak passwords list
checking passwords
Starting 1 threads
JRULES  OK [OPEN]
SNPM    OK [OPEN]
SNPW    OK [OPEN]
MIDOFF  OK [OPEN]
WAREHOUSE has weak password BUSINESS [OPEN]
DISTRIBUTE has weak password HOUSE [OPEN]
MDSTATUS has weak password MDSTATUS [OPEN]
QC      OK [OPEN]
DASHBOARD    OK [OPEN]
SYS_USER      OK [OPEN]
DEPLOYER  OK [OPEN]
TSMSYS  OK [LOCKED]
DBSNMP  OK [OPEN]
EXFSYS  OK [LOCKED]
XDB     OK [LOCKED]
SYSMAN  OK [LOCKED]
MGMT_VIEW       OK [OPEN]
SYS     OK [OPEN]
SYSTEM  OK [OPEN]
OUTLN   OK [LOCKED]

Done. Summary:
  Passwords checked      : 43241
  Weak passwords found   : 3
  Elapsed time (min:sec) : 0:00
  Passwords / second     : 43241
```

4.2.2. Mots de passe par défaut

Type	2 (intrusion)
Contrôle	Validation régulière à l'aide d'outils Backtrack (checkpwd + default_passwords.txt), basé sur la comparaison de chaînes hachées. Contrôle périodiques en brut force
Mesure	Modification des mots de passe par la DBA. Eviter de rejouer catproc.sql (mise à défaut de mors de passe)

4.2.3 Mots de passe codé en dur

Type	2 (intrusion)
Contrôle	Via trigger sur connexion (cf. ci-dessous), en s'assurant que l'applicatif appelant est le bon.
Mesure	Externalisation du mot de passe crypté dans fichiers de paramètres Documentation pour chacune des data sources Modification chronique de ces mots de passe

Voici un exemple de déclencheur sur connexion qui permet de n'autoriser l'utilisation de certains applicatifs qu'à certains comptes.

```
create or replace TRIGGER "SYS"."BLOCK_USER_ACCESS"
AFTER LOGON ON DATABASE
DECLARE
  v_prog sys.v_$session.program%TYPE;
  v_dbuser sys.v_$session.username%TYPE;
  v_osuser sys.v_$session.osuser%TYPE;
  v_db sys.v_$database.name%TYPE;

/*
 * Auteur : Fabien Celaia
 * Date   : 7 juillet 2006
 * Desc.  : Empeche l'utilisation de certains logins
avec certains applicatifs
 */

BEGIN
  /* Récupération des informations utiles dans
v$session */

  SELECT upper(program), upper(username), upper(osuser)
  INTO   v_prog, v_dbuser, v_osuser
  FROM sys.v_$session
  WHERE  audsid = USERENV('SESSIONID')
    AND  audsid != 0  -- N'impacte pas la connexion SYS
    AND  rownum = 1;  -- Gestion du parallélisme (memes
AUDSID)

   SELECT upper(name)
   INTO   v_db
   from v$database ;

  /* Utilisation à proscrire */
  IF v_dbuser NOT IN ('SYSTEM', 'SYS', 'STREAM',
'DBSNMP')
     AND coalesce(v_osuser,'SYSTEM') not in ('SYSTEM')
     AND (v_prog ='TOAD.EXE'
       OR v_prog LIKE '%ACCESS.EXE'
       OR v_prog LIKE '%SQIREL%'
       OR v_prog LIKE 'SQLPLUS%'
       OR v_prog LIKE 'SQL DEV%'
       OR v_prog LIKE '%MSQ%.EXE')
  THEN
    RAISE_APPLICATION_ERROR(-20000,  v_osuser ||'
n''est pas autorisé à utiliser  '|| v_prog ||' sur
l''environnement '||v_db) ;
  END IF;

  EXCEPTION
    WHEN NO_DATA_FOUND THEN NULL;
END;
```

Il s'agit là d'une sécurité toute relative compte tenu du fait qu'il est aisé de modifier le nom du programme appelant via PL-SQL, que ce trigger ne se déclenche pas avec sysdba... et pour de nombreuses autres raisons exposées dans la Faq Oracle (Lien44).

4.2.4. Spoliation temporaire de mot de passe

Avec des droits DBA, il est tout à fait aisé de s'approprier l'identité d'un utilisateur spécifique de manière temporaire:

Le mot de passe hashé est un md5 de la concaténation du login, du mot de passe et d'un bout de chaîne. Il est donc invariable d'une instance à une autre pour un login/mot de passe donné. De plus, le mot de passé hashé de scott/tigger sera le même que celui de sco/tttiger.

Dès la version 10g, le mot de passe devient sensitif, ce qui n'était pas le cas au préalable. Mixez donc majusculeset minuscules afin de complexifier la tâche aux outils de brute force.

Dans un premier temps, on relève son mot de passe crypté, connecté en tant que dba.

```
SQL> connect system/motdepasse
Connecté

SQL> select password from dba_users where
username='MONCOCO'

PASSWORD
------------------------------
F894765434402B67
```

Il est ensuite aisé de changer son mot de passe, puis de se connecter sous son profile

```
SQL> alter user MONCOCO identified by MonPwd

User altered

SQL> connect MONCOCO/MonPwd
Connecté
```

Notre méfait accompli, il ne nous reste plus qu'à remettre l'ancien mot de passe

```
SQL> connect system/motdepasse
Connecté

SQL> alter user MONCOCO identified by values
'F894765434402B67'
```

Il est possible de limiter ce genre d'intrusion, en empêchant par exemple la modification d'un mot de passe via trigger sur ALTER, mais cela ne limitera pas un compte de type sysdba.

4.3. Intrusion via système d'exploitation

Une des principale faille de la sécurité Oracle est qu'il n'est quasi pas possible d'empêcher à un administrateur du système d'exploitation de se connecter en tant qu'admnistrateur de la base de données.

Sous Windows, un administrateur est tout à fait en mesure de se mettre dans le rôle système oradba. Sous Unix, idem pour le rôle, et de manière encore plus aisée, il sera en mesure d'exécuter un sudo - oracle.

Cette capacité à se connecter sur une instance Oracle sans mot de passe, directement en sqlplus / as sysdba permet à quiconque qui aurait des droits d'administration sur la machine de se connecter en tant que superuser sur Oracle.

Cela peut aussi être considéré comme un avantage lors du départ innopiné ou du décès du DBA !

Type	2 (intrusion)
Contrôle	Eviter, pour les DBA aussi, de se connecter en tant que sys
Mesure	Comptes DBA spécifiques et nommés, audit fait (par exemple, c'est le cas avec Oracle Grid Control)

4.4. Intrusion par application frontale

Que ce soit via SQLDeveloper ou via Toad, les configurations de connexion sont stockées sur le PC client du développeur/DBA. Bien que ces 2 applicatifs stockent les mots de passe en cryptage EAS, il est possible, au travers d'un disque partagé par exemple, d'aller " vampiriser " le fichier de configuration afin de s'approprier les accès excessifs, sans pour cela connaître les mots de passe utilisés.

4.4.1. Informations de connexion stockées en local

Les fichiers sensibles sont

- pour SQL Developer : ...\sqldeveloper\jdev\system\oracle.onlinedb.11.0.0.37.42\IDE*.*
- pour Toad : ...\Quest Software\Toad for Oracle\User Files*.*
- pour sqlplus, sqlplusw : toute sorte de raccourci dans environnement Windows

Type	2 (intrusion)
Contrôle	Validation régulière à l'aide d'outils Backtrack (checkpwd + default_passwords.txt), basé sur la comparaison de chaînes hachées.
Mesure	Suppression ou limitation du partage C$ sur les postes. Interdiction faite aux utilisateurs d'utiliser toute forme de sauvegarde de mot de passe dans les connexions.

4.4.2 Clients non patchés / non sécurisés

Il est nécessaire de maîtriser l'installation des clients Oracle, ceux-ci étant aussi sensibles aux failles de sécurité. Bien souvent, l'installation des bases et de leurs binaires sont dévolues au DBA, mais les installations clientes Oracle sont souvent faites "à la sauvage", via des packages ou un département "bureautique" peu sensible aux soucis sécuritaires qui nous occupent.

Type	3 (injection SQL)
Contrôle	Eviter toute installation cliente sauvage
Mesure	Catalogage de toute installation cliente Patch sécurité à installer côté client aussi

Type	4 (corruption)
Contrôle	Installation des clients correcte, variables d'environnement adéquates (NLS) Pas de caractères étranges saisis
Mesure	Reprise en main des installations clientes Configuration adéquate des clients Oracle, et plus particulièrement des variables NLS

4.4.3. Fichiers autoexécutables

Du code SQL néfaste peut être insidieusement placé sur le disque d'un super-utilisateur et autoexécuté au démarrage de sqlplus. Ceci se fait dans les fameux fichiers glogin.sql et login.sql, situés soit dans Oracle_home\bin, soit dans Oracle_home\sqlplus\admin.

Type	3 (injection SQL)
Contrôle	S'assurer que les fichiers login.sql et glogin.sql ne se trouvent pas dans le répertorie ${ORACLE_HOME}/bin des clients
Mesure	Démarrer sqlplus à l'aide d'un fichier script/batch nettoyant au préalable tout glogin.sql ou login.sql du répertoire incriminé.

4.5. Intrusion par sniffage du processus d'écoute

Le listener d'Oracle peut être détourné à des fins néfastes. Il permet de sniffer le protocole de transfert Oracle non crypté. Les requêtes, les données passent en clair au travers de la trame. Plusieurs solutions peuvent être mise sur pied afin de pallier à ce problème. Il convient aussi de spécifier au listener de quelles bases il est censé géré la communication.

Type	1 (déni de service)
Contrôle	Check du listener.ora afin de s'assurer de sa bonne configuration
Mesure	- en version pré-10, mise en place d'un mot de passe pour le listener - dès la version 10, pas de mot de passe : la mise en place de ce dernier permettrait un accès distant beaucoup plus dangereux

Type	2 (intrusion)
Contrôle	Check du listener.ora, de $TNS_ADMIN afin qu'ils ne comportent pas de redirection
Mesure	- en version pré-10, mise en place d'un mot de passe pour le listener - dès la version 10, pas de mot de passe : la mise en place de ce dernier permettrait un accès distant beaucoup plus dangereux, et il n'est pas supporté dans un environnement RAC tel que le nôtre. - acquisition d'Oracle Advance Security Option (~ 10'000 USD/CPU) incluant Network Security - Cryptage des données sensibles sur la base et décryptage sur le client

Type	3 (injection SQL)
Contrôle	Check du listener.ora, du tnsnames.ora, de $TNS_ADMIN afin qu'ils ne comportent pas de configuration extproc
Mesure	Suppression de la possibilité d'utiliser des procédures externes

4.6. Scannage de port

Pour les applications n'étant pas orientées extranet, le risque de subir ce genre d'attaque de l'extérieur est plus limité. Cependant, il existe en interne, mais est aisément détectable, les outils de scannage (nmap, amap, etc.) étant perçus comme des attaquants par les outils réseau. Le risque existe néanmoins puisque ce type

d'attaque est en mesure de faire tomber certains modules Oracle (Notification Delivery Service).

Type	1 (déni de service)
Contrôle	Par outils réseau interne (IDS) détectant ce type de scannage.
Mesure	Audit réseau

4.7. Intrusion par binaires non sécurisés

Les failles de sécurité n'étant fixées que dans les dernières versions, les anciens binaires Oracle sont plus sensibles. Il convient donc de supprimer tout ancienne version de binaire.

Type	3 (intrusion)
Contrôle	Suppression de dumpsga (permettant de flusher la mémoire sur disque). Cryptage des datafiles
Mesure	rm $ORACLE_HOME/bin/dumpsga Utilisation de Transparent Data Encryption d'Oracle

Type	4 (injection SQL)
Contrôle	Suppression des binaires non patchés
Mesure	rm $ORACLE_HOME/bin/*.bak rm $ORACLE_HOME/bin/*.0

4.8. Injection SQL

Certains modules Oracle offrent des possibilités d'intrusion SQL. Il est donc nécessaire de réduire au maximum les droits (par défaut ALL to PUBLIC) sur ces modules. Ceux-ci ne peuvent pas être purement et simplement supprimés, car appartiennent au noyau Oracle.

```
revoke execute on sys.dbms_obfuscation_toolkit from
public ;
grant execute on sys.dbms_obfuscation_toolkit to sysman
;
grant execute on sys.dbms_obfuscation_toolkit to dbsnmp
;
grant execute on sys.dbms_obfuscation_toolkit to wksys
;
Grant execute on sys.dbms_obfuscation_toolkit to
flows_030000 ;
revoke execute on sys.utl_tcp from public ;
grant execute on sys.utl_tcp  to sysman ;
revoke execute on sys.utl_http from public ;
grant execute on sys. utl_http  to ordplugins ;
revoke execute on sys.utl_smtp from public ;
grant execute on sys.utl_smtp  to sysman ;
revoke execute on sys.utl_file from public ;
grant execute on sys. utl_file  to ordplugins ;
Grant execute on sys.utl_file to mitg_rml ;
Grant execute on sys.utl_file to mitg ;
grant execute on sys. utl_file  to xdb ;
grant execute on sys. utl_file  to sysman ;
grant execute on sys. utl_file  to dmsys ;
revoke execute on sys.dbms_lob from public ;
grant execute on sys.dbms_lob  to xdb ;
```

```
Grant execute on sys.dbms_LOB tu ctxsys ;
Grant execute on sys.dbms_lob to flows_030000 ;
Grant execute on sys.dbms_lob to mitg_rml ;
Grant execute on sys.dbms_lob to mitg ;
grant execute on sys.dbms_lob  to dmsys ;
grant execute on sys.dbms_lob  to exfsys ;
grant execute on sys.dbms_lob  to dba ;
grant execute on sys.dbms_lob  to ctxsys ;
grant execute on sys.dbms_lob  to mdsys ;
grant execute on sys.dbms_lob  to ordsys ;
grant execute on sys.dbms_lob  to wksys ;
grant execute on sys.dbms_lob  to olapsys ;
grant execute on sys.dbms_lob  to ordplugins ;
revoke execute on sys.dbms_job from public ;
grant execute on sys.dbms_job  to dbsnmp ;
grant execute on sys.dbms_job  to wksys ;
grant execute on sys.dbms_job  to exfsys ;
Grant execute on sys.dbms_job to flows_030000 ;
revoke execute on sys.utl_inaddrm from public ;
revoke execute on sys.dbms_export_extension from public
;
grant execute on sys.dbms_export_extension to dba ;
grant execute on sys.dbms_export_extension to system ;
revoke execute on sys.dbms_backup_restore from public ;
revoke execute on sys.dbms_sql  from public ;
grant execute on sys.dbms_sql  to sysman ;
grant execute on sys.dbms_sql  to system ;
grant execute on sys.dbms_sql  to xdb ;
grant execute on sys.dbms_sql  to exfsys ;
Grant execute on sys.dbms_sql to flows_030000 ;
grant execute on sys.dbms_sql  to dmsys ;
Grant execute on sys.dbms_sql to mitg_rml ;
Grant execute on sys.dbms_sql to mitg ;
grant execute on sys.dbms_sql  to mdsys ;
grant execute on sys.dbms_sql  to olapsys ;
grant execute on sys.dbms_sql to ctxsys ;
grant execute on sys.dbms_sql  to oracle_ocm ;
revoke execute on sys.dbms_ldap  from public ;
revoke execute on sys.dbms_advisor from public ;
Grant execute on sys.dbms_ldap to flows_030000 ;
GRANT EXECUTE ON  SYS.DBMS_LDAP TO sysman ;
```

Sans oublier pour chaque schéma pouvant être importé:
grant execute on sys.dbms_export_extension to MonSchema ;

Pour diverses raisons (installation de modules Oracle, tel que la dbconsole par exemple, il est souhaitable d'avoir les droits initiaux sur ces modules. Voici donc les requêtes SML de rétro conversion :

```
grant all on sys.dbms_obfuscation_toolkit TO PUBLIC ;
grant all on sys.utl_tcp TO PUBLIC ;
grant all on sys.utl_http TO PUBLIC ;
grant all on sys.utl_smtp TO PUBLIC ;
grant all on sys.utl_file TO PUBLIC ;
grant all on sys.dbms_lob TO PUBLIC ;
grant all on sys.dbms_job TO PUBLIC ;
grant all on sys.utl_inaddrm TO PUBLIC ;
grant all on sys.dbms_export_extension TO PUBLIC ;
grant all on sys.dbms_backup_restore TO PUBLIC ;
grant all on sys.dbms_sql  TO PUBLIC ;
grant all on sys.dbms_ldap  TO PUBLIC ;
grant all on sys.dbms_advisor TO PUBLIC ;
```

Retrouvez l'article de Fabien Celaia en ligne avec encore d'autres dangers et mesures de corrections : Lien45

Windows/ Hardware

Les derniers tutoriels et articles

Comment choisir sa configuration PC ?

Comment choisir une configuration adaptée à ses besoins ? Voici quelques échantillons de ce que je vous recommande pour diverses applications, de la bureautique au calcul intensif, en passant naturellement par le jeu.

1. Quelques règles de base

Il existe toujours des règles de base à suivre, en voici quelques unes.

- Choisir une alimentation de marque avec le label 80Plus. Pourquoi ? Ces alimentations sont fiables, et surtout elles consomment peu. Elles coûtent un peu plus cher, mais elles vivent plus longtemps que les noname et impactent moins votre facture d'électricité.
- Choisir un maximum d'éléments passifs dans sa configuration a l'avantage de limiter le bruit produit par l'ensemble, mais aussi indique que la consommation des différents éléments est relativement plus faible que des d'autres éléments actifs.
- Les disques durs se valent presque tous...
- Pour l'instant, AMD a le meilleur rapport puissance/prix dans l'entrée de gamme, dans le milieu de gamme, Intel reprend le dessus et écrase AMD dans le haut de gamme.
- Dans les différentes configurations, je n'inclus pas le prix de la licence Windows, même si j'indiquerai le type d'OS à privilégier.

2. Configuration bureautique

Orienté vers le traitement de texte, la navigation Internet, ... ce PC n'a pas besoin de beaucoup. L'OS que je recommande pour ce type d'appareil est un XP familial.

- Processeur : AMD Sempron 3200+ sur support AM2 (35€)
- Carte mère : Gigabyte GA-M61P-S3 (chipset graphique intégré) (74€)
- Mémoire : Corsair Value Select Kit 1Go (2*512Mo, DDR2-533) (63€)
- Disque dur : Samsung Pinpoint S HD161HJ 160Go (53€)
- Lecteur/graveur DVD : quelconque (max 50€)
- Alimentation : Fortron Green 400W (60€)
- Boîtier : Antec NSK 4000 EU (40€)

Pour environ 400€, sans écran, vous obtenez un PC complet (pensez tout de même au clavier et à la souris), économique et "écologique" (alimentation à haut rendement, pas de carte graphique supplémentaire, ...).

En ce qui concerne les écrans, je conseille un Samsung SyncMaster 710N (175€) ou un LG L1718S-SN (170€), des écrans LCD 17", donc pas trop petits et pas trop grands, consommant peu ; pour de la bureautique, cela sert à rien d'avoir plus grand et ça consomme moins.

3. Configuration tout-terrain

Ce que j'appelle configuration tout-terrain, c'est une configuration silencieuse mais performante pour jouer sans trop se ruiner, pour faire de la MAO semi-pro (Musique Assistée par Ordinateur), et donc aussi du développement logiciel, ... Pour ce genre de PC, un XP pro ou un Linux est conseillé.

- Processeur : Intel Core2Duo E6320 (160€)
- Ventirad : Noctua NH-U12F (60€)
- Carte mère : Gigabyte GA-965P-DS4 (GeForce 8600) (161€)
- Mémoire : Crucial 2 Go (2*1 Go, DDR2-800) CL5 - CT2KIT12864AA80E (129€)
- Carte graphique : Gigabyte GV-NX86T256D (143€)
- Disque dur : Hitachi T7K500 (320 Go) (95€)
- Graveur DVD : quelconque (max 50€)
- Alimentation : fournie dans le boîtier
- Boîtier : Antec P150 (140€)

Pour 940€, un PC avec lequel on peut faire presque tout, jouer aux jeux récents ou mixer le dernier album de son groupe. Le ventirad n'est pas indispensable, mais avoir une bonne dissipation limite la vitesse des ventilateurs et donc leur bruit. L'alimentation fournie dans le boîtier P150 est à haut rendement (fabriquée par Seasonic) et silencieuse (après des débuts difficiles...) et les ventilateurs internes possèdent plusieurs vitesses de rotation. Enfin, pas de ventilateur sur la carte mère et la carte graphique. Il est aussi possible de ne pas prendre le ventirad Noctua si le prix total est trop cher.

Pour une application plus professionnelle, je recommande un disque dur principal de 80Go (Hitachi) pour l'OS. Si vous faites de la MAO, rajoutez un troisième disque dur pour les samples.

En ce qui concerne l'écran, les Belinea 10 20 35 W (359€) ou LG Flatron L2000C (300€) sont très bons, grands (20") et économiques. Le LG est plus orienté gamer tandis que le Belinea est plus orienté application professionnelle.

4. Configuration hardcore gamer

Le principe de base de cette configuration est de prendre le plus ou presque et le plus rapide... L'OS est un XP quelconque.

- Processeur : Intel Core2Duo E6700 (240€)

- Ventirad : Noctua NH-U12F (60€)
- Carte mère : Asus P5W DH Deluxe (180€)
- Mémoire : Crucial Ballistix Tracer 2 Go (2*1 Go, DDR2-800) 4-4-4-12 - BL2KIT12864AL804 (170€)
- Carte graphique : ASUSTeK EN8800GTS (308€)
- Disque dur : Western Digital Raptor 74 Go (155€) et Western Digital Caviar SE16 400 Go WD4000KS (110€)
- Graveur DVD : quelconque (max 50€)
- Alimentation : fournie dans le boîtier
- Boîtier : Antec P150 (140€)
- Ecran : LG Flatron L2000C (300€) (20")

Avec un total de 1710€, cette configuration devrait en faire rêver plus d'un. Question tout de même, est-ce que l'alim de seulement 430W survit ? Oui, naturellement, même si la carte graphique dépasse les 200W, il reste de la marge, le reste ne chauffant que très peu. Même en SLI, la configuration tient le coup !

Si vous achetez une telle configuration, il est important de paramétrer correctement la carte mère pour tirer parti au maximum des capacités des différents éléments.

5. Configuration calcul intensif

Ici, je vais donner un patron de configuration type plutôt qu'une vraie configuration, car une application CAO (Conception Assistée par Ordinateur) utilisera une carte graphique professionnelle (Quadro ou autre), ce qui n'est pas le cas du calcul scientifique (à moins de faire du GPGPU).

- Processeur : Intel Core2Quad Q6600 (880€)
- Ventirad : Noctua NH-U12F (60€)
- Carte mère : Asus P5W DH Deluxe (180€)
- Corsair XMS2 DHX TWIN2X4096-6400C5DHX 2*2Go (299€)
- Carte graphique : Gigabyte GV-NX86T256D (143€)
- Disque dur : Hitachi 7K80 (80 Go) (41€) comme disque pour l'OS, Hitachi T7K500 (320 Go) pour les données et/ou les programmes (95€)
- Graveur DVD : quelconque (max 50€)
- Alimentation : Seasonic S12-500 (107€)
- Boîtier : Antec P180 (124€)

2300€ environ pour une telle configuration, à modifier selon les besoins (plus de RAM, une meilleure carte graphique, plus d'espace disque, ...)

Retrouvez l'article complet de Miles en ligne : Lien46

La séquence de démarrage d'un PC

Pour savoir d'où vient une erreur lors du lancement de votre ordinateur, il peut être très utile de savoir comment se passe le démarrage de votre pc et quelle est la signification des messages qui peuvent éventuellement arriver. C'est cela que nous allons voir dans cet article.

1. Introduction

La séquence de démarrage c'est toutes les étapes qui vont être exécutées dès le moment ou vous allez démarrer votre ordinateur. Il y a plusieurs éléments qui entrent en jeu durant cette séquence. A quoi sert de connaître cette séquence ? Tout simplement en cas de problème, à mieux identifier celui-ci et à mieux le résoudre. Et aussi bien entendu à mieux connaître votre ordinateur.

La séquence que je vais décrire ici est la séquence standard pour un système fonctionnant avec un système d'exploitation Windows NT.

2. La séquence

Pour commencer, dès le moment où vous allez appuyer sur le bouton de mise sous tension de votre PC, une impulsion électrique va être envoyée à l'alimentation depuis la carte mère. Laquelle va ensuite produire du courant, courant qui va allumer le Bios.

2.1. Le BIOS ou séquence POST

Le BIOS s'occupe de tester et d'initialiser tous les matériels. On appelle aussi cette partie la séquence POST (Power On Self Test) ou encore séquence préboot. C'est la séquence durant laquelle tous les composants vont être testés de même que leur compatibilité. Si la séquence POST ne passe pas, le système n'ira pas plus loin et votre OS ne sera pas lancé. Il va commencer par contrôler le bus système et va vérifier ensuite tous les connecteurs d'extension. Il va continuer en vérifiant la mémoire de la carte graphique et les signaux commandant l'affichage. Ensuite, il va interroger le BIOS de la carte vidéo et ajouter son code de reconnaissance. C'est à partir de ce moment-là que les premiers affichages arrivent à l'écran. Il va tester la RAM, pour cela, il tente une écriture sur chaque zone mémoire et tente de lire ensuite pour les comparer à ce qu'il a écrit. Il vérifie si le clavier et la souris sont bien connectés. Ensuite, il envoie des signaux à tous les périphériques de stockage (disquette, cd, HDD, USB, …) pour définir quels sont les différents lecteurs. Tous les résultats sont comparés sur le CMOS, ce qui permet au BIOS de savoir si la configuration matérielle a changé depuis le dernier démarrage ou pas. Ensuite, il intègre les identifiants de tous les composants ayant un BIOS.

Ensuite, les tests matériels validés, il va tenter d'amorcer en mémoire le secteur d'amorce principal du disque dur aussi appelé MBR.

Il y a pas mal d'erreurs qui peuvent se produire durant cette phase, elles sont le plus souvent d'ordre matériel, par exemple une barrette de RAM mal branchée ou un composant manquant ou encore une incompatibilité entre 2 matériels. Ces erreurs sont indépendantes du système d'exploitation.

2.2. Le MBR

Le MBR (Master Boot Record) ou table de partition en français, permet de trouver la partition active du disque. Une fois que cette partition est identifiée, le MBR va charger le secteur de boot correspondant et transférer ensuite l'exécution à ce dernier.

Les erreurs pouvant arriver à ce stade du démarrage sont souvent d'ordre de stockage. C'est-à-dire qu'il peut y avoir plusieurs partitions actives, ou aucun support de stockage valable. Ou alors, il peut arriver que la table de partition soit altérée.

2.3. Le secteur de boot

Une fois que le MBR lui a donné la main, le secteur de boot va charger les 15 secteurs qui le suivent sur le disque et va ensuite transférer le contrôle à un programme présent sur ces secteurs. Ces 15 premiers secteurs sont appelés " Boostrap Code " et s'occupent de localiser puis de transférer l'exécution au fichier NTLDR.

Les erreurs qui peuvent arriver à ce niveau sont encore une fois surtout des problèmes hardware. C'est-à-dire que par exemple un des secteurs qu'il doit charger est manquant. Ou alors que le disque sur lequel on démarre n'a pas de NTLDR, donc on ne peut pas booter dessus. Ou alors, il peut arriver qu'il y aie un problème avec le fichier NTLDR.

2.4. NTLDR

On va maintenant passer sur le NTLDR, qui marque cette fois la première partie de l'exécution de Windows. C'est le chargeur d'amorçage de Windows. C'est lui qui va savoir quels windows sont installés et lequel il faut lancer. Il commence par charger les pilotes du système de fichier approprié. Ensuite, en fonction du fichier Boot.ini, il va définir quels sont les systèmes d'exploitations qu'il peut lancer et s'il y en a plusieurs, il va les afficher à l'écran et demander à l'utilisateur d'en choisir un. Il charge le programme NTDETECT qui va ensuite détecter le matériel du pc. Il charge plusieurs dll qui vont permettre d'effectuer la suite du travail. Il charge la majorité de la base de registre (le reste étant chargé plus tard par le système d'exploitation). Et enfin, il donne le contrôle à NTOSKRNL.exe.

Les problèmes qui peuvent arriver ici sont surtout des problèmes liés aux fichiers qui doivent être lancés, par exemple un fichier qui manque ou alors un problème d'accès à un des fichiers.

2.5. NTOSKRNL.exe

Nous voilà à la fin de la séquence de démarrage du PC, cette fois, le noyau NT va se lancer définitivement et va charger le programme de logon et nous allons nous retrouver sur notre bon vieux Windows.

3. Bips et alertes

Il est possible que pendant la séquence de démarrage, votre ordinateur marque un temps d'arrêt. Il se peut aussi qu'il émette des bips ce qui est en fait son seul langage pour nous avertir. Les bips qui sont émis dépendent entièrement du BIOS. Après que l'affichage se soit initialisé, il se peut aussi qu'il affiche des messages d'erreurs.

3.1. Bips

A cause de la multitude de type de Bios qui existe, il n'y a pas de standards au niveau des bips et il est vrai qu'il n'est pas facile de s'y retrouver. Je ne vais couvrir ici que les trois principaux BIOS, Phoenix, Award et AMI. Plusieurs autres types de BIOS ont ensuite pris un de ces constructeurs comme exemple, mais il est vrai que cela ne couvre pas tous les bips possibles pour tous les constructeurs.

Si vous avez un autre type de Bios, je vous invite à aller chercher la signification des bips de démarrage soit directement sur le site du constructeur soit sur le site Bios Central qui recense les bips de démarrage pour plusieurs constructeurs.

3.2. Messages

Une fois que l'affichage est chargé, le BIOS peut aussi vous afficher des messages d'erreurs, ce qui est souvent plus explicite que des bips. Encore une fois, il n'y a pas de standard au niveau de ces messages d'erreurs, mais cette fois, c'est tout de suite plus compréhensible, car souvent il suffit de lire le message pour comprendre l'erreur. Je ne vais cette fois encore détailler que les messages d'erreurs des BIOS Phoenix, AMI et Award. Si vous avez besoin d'informations sur un autre type de BIOS, je vous invite encore une fois à consulter Bios Central. Certains messages ne sont pas des messages d'erreurs mais simplement des messages d'informations.

Chaque fois que le message " Run Setup " apparaît, cela veut dire qu'il faut reconfigurer quelque chose dans le BIOS en fonction du message.

4. Conclusion

Voilà, vous savez maintenant comment se passe le démarrage de votre PC et vous connaissez aussi la signification des bips et des messages d'erreurs pouvant survenir durant ce démarrage. J'espère que cela pourra vous aider à régler certains problèmes.

Retrouvez l'article complet de Baptiste Wicht en ligne avec la signification des différents messages et alertes : Lien47

Les livres Windows

Windows Vista : le Magnum

Profitez pleinement de toutes les innovations de Windows Vista ! Avec cet ouvrage, vous apprendrez à utiliser Windows Vista et découvrirez les immenses possibilités de ce nouveau système d'exploitation. En intégrant le meilleur des dernières technologies, Windows Vista marque une véritable révolution. Au fil des pages : Vous verrez pourquoi Vista améliore la sécurité dans l'utilisation courante d'Internet et, d'une façon plus générale, dans l'utilisation quotidienne du système d'exploitation. Vous apprendrez à utiliser le nouvel outil d'indexation et de recherche pour accéder en un tournemain à vos programmes et à vos données. Vous découvrirez les nouveaux programmes, outils et accessoires inclus dans Windows Vista. Vous apprécierez les outils en ligne de la gomme Live de Microsoft. Vous apprendrez à installer et à configurer le matériel qui vous permettra d'accéder à Internet. Si vous disposez de plusieurs ordinateurs, vous saurez comment les paramétrer pour partager votre connexion Internet, vos fichiers et vos périphériques. Si vous voulez aller plus loin, vous verrez comment manipuler le Registre, comment automatiser les tâches que vous effectuez le plus fréquemment, comment créer vos propres gadgets, comment installer puis utiliser PowerShell, l'alternative évoluée de la fenêtre MS-DOS, comment utiliser le langage HTML pour créer votre propre site Web et bien d'autres choses encore. Enfin, même si Windows Vista est un système sécurisé et peu enclin à des défaillances, vous verrez comment entretenir votre ordinateur pour le garder ou mieux de sa forme, et comment résoudre la plupart des problèmes matériels et logiciels.

Critique du livre par la rédaction (Louis-Guillaume Morand)

J'ai toujours trouvé un peu prétentieux de la part d'un éditeur de nommer un article Bible ou Magnum ou autre, mais pour la première fois je me dois d'avouer que ce livre a quelque chose d'autre. Tout d'abord, il est bien supérieur en qualité et en contenu aux autres livres sur Windows Vista qu'il m'a été donné de lire. Je me permets même pour la première fois de mettre la note maximale à un livre pour avoir réussir à satisfaire ma curiosité jamais rassasiée.
Michel MARTIN, déjà auteur de très nombreux livres arrive à dessiner un chemin pédagogique dans l'apprentissage de ce nouveau système, tout en prenant le soin de n'épargner aucun composant aussi futile puisse-t-il paraitre.

Je ne parlerai pas de la présentation du système qui ne démarque des autres légèrement de par sa qualité d'écriture, je parlerai directement des concepts avancés qu'il aborde.
Ainsi c'est le premier livre que je rencontre qui contient une partie résolution de problèmes aussi avancée. Non seulement vous apprendrez à résoudre les problèmes de drivers souvent rencontrés (son, vidéo, etc.) mais vous apprendrez surtout comment gérer tout type de problème avec Windows Vista et à trouver une solution par vous même en analysant le journal d'évènement ou en utilisant des logiciels tiers comme TweakVI.
Le livre contient également un chapitre entier au démarrage de Windows Vista, de la configuration du BIOS et les éventuels problèmes qu'il peut engendrer, jusqu'aux problèmes de périphériques comme les modems ou autre.

Enfin, pour satisfaire les lecteurs les plus dégourdis, Michel Martin propose des introductions complètes à l'utilisation du registre, la création de Gadget (pour la SideBar) ou la création de scripts PowerShell. Pour chacun de ses concepts, l'auteur propose également une sélection de liens "Pour aller plus loin" vous permettant d'avoir rapidement accès aux informations avancées.

Au final que vous soyez un grand débutant de l'informatique, un néophyte avec Vista ou même un utilisateur averti, ce Magnum pourra dans de nombreux cas répondre à vos interrogations et cela très rapidement. **Un livre de chevet pour tout utilisateur de Windows Vista!!!**

Retrouvez ce livre sur la rubrique Windows : Lien48

Windows Vista : Installation, configuration et administration

Ce livre est destiné aux informaticiens ou utilisateurs avertis, soucieux de bien connaître ce système d'exploitation. Il détaille l'installation, la configuration et l'administration de Windows Vista sur un poste de travail, dans un contexte de production (Editions Professionnelle, Entreprise et Intégrale). Vous y trouverez tous les détails sur l'installation du système, des périphériques des pilotes, sur la gestion de l'environnement de travail, la gestion du réseau, des utilisateurs, des groupes, des ressources, sur la gestion de la sécurité et sur les outils d'analyse et de dépannage. Les nouveautés Vista, qui améliorent des fonctionnalités existant déjà sous XP ou Windows 2003 sont détaillées et replacées dans leur contexte et les nouvelles fonctionnalités sont décrites et mises en œuvre : le volet Windows et l'interface AERO (l'environnement de travail), Xlmage (l'outil destiné à faciliter les déploiements) Windows SideShow (la prise en charge d'affichages auxiliaires), Windows Desktop Search (le moteur de recherche accessible dès le menu démarrer), le centre de Mobilité (fonctionnalité destinée à favoriser le travail des utilisateurs nomades), le pare-feu Windows (fonctionnalités avancées du pare-feu), Windows Defender (le logiciel anti-spyware), le centre de Sécurité (le centre de la sécurité, la console de gestion de la sécurité, les modèles de sécurité), BitLocker Drive Encryptlon (le cryptage hautement sécurisé des données sur un disque), le centre Réseau (centralisation de la gestion des connexions réseau), le contrôle parental (configuration précise du contrôle parental et suivi de l'activité) User Account Control (niveau supplémentaire dans les droits de l'utilisateur), la console de gestion de l'impression (console de gestion des imprimantes)...

Critique du livre par la rédaction (Nicolas Vallée)

En 450 pages distribuées en 9 chapitres l'auteur amène tout un chacun à pouvoir installer, paramétrer, dépanner son système, son environnement de travail, son réseau, ... de manière simple.

Chaque étape est illustrée de si nombreuses copies d'écran qu'il est quasiment impossible de faire la moindre erreur. Après l'installation du poste de travail sont abordées la configuration des groupes d'utilisateurs, la gestion des ressources et de la sécurité.

Vient ensuite la présentation de la boîte à outils Vista : les outils de diagnostics et de sauvegarde.

En résumé un livre complet, accessible à tous, le genre de guide indispensable, qui sans vouloir rivaliser avec un manuel de référence, permet de gérer l'ensemble d'une structure bureautique et a fortiori une machine isolée.
A moins de trente euros, c'est un investissement rentable, une solution efficace après l'achat de Vista.

Attention, cet ouvrage ne se veut pas un membre de la collection "Informatique pour les nuls", il est réellement utilisable et à la portée de tous, sans pour autant sombrer dans la catégorie des ouvrages pour experts. Toutefois, les administrateurs de réseaux d'entreprises n'y trouveront pas toujours toutes les informations qu'ils souhaiteraient.

Retrouvez ce livre sur la rubrique Windows : Lien49

Les derniers tutoriels et articles

À la découverte de QuartzComposer

Certains développeurs excellent dans l'art de réaliser des effets graphiques impressionnants. Apple a pensé à tous les autres en livrant QuartzComposer avec Tiger, la dernière version de MacOS X.

1. Introduction

Le DVD d'installation de MacOS X 10.4 contient un grand nombre d'outils de développement de qualité. La plupart, comme Xcode 2.0, ne sont que des évolutions de ceux livrés avec les versions précédentes du système. Néanmoins, pour marquer l'introduction de la nouvelle bibliothèque Core Image, Apple a ajouté deux outils extrêmement puissants, Core Image Fun House et QuartzComposer. Le premier permet d'appliquer tous les effets de Core Image sur une image et d'en modifier les paramètres. Vous pouvez ainsi aisément tester des combinaisons avant de les intégrer dans votre code source. QuartzComposer fonctionne un peu sur le même principe mais va beaucoup plus loin. Cet outil vous permet de créer des scènes graphiques interactives et dynamiques. Vous pouvez utiliser tous les effets de Core Image et Core Video, mais également combiner des Image Units, des éléments OpenGL ou QuickTime, parcourir des flux RSS, utiliser des opérations Quartz 2D et enfin intégrer les services audio MIDI. Le résultat peut être joué comme une animation autonome, être utilisé comme économiseur d'écran on se voir intégrer à n'importe quelle application Cocoa par l'entremise d'une API très simple. Apple a même prévu la possibilité de lier facilement des contrôles Cocoa aux paramètres des scènes QuartzComposer. Sachez enfin que cette application n'est pas une véritable nouveauté puisqu'une version beaucoup plus simple intitulée PixelShox Studio avait été créée en 2002 par Pierre-Olivier Latour qui a ensuite participé à l'élaboration de QuartzComposer. Avant de continuer, assurez-vous d'avoir installé Xcode 2.0 et les outils de développement depuis le DVD de Tiger et lancez QuartzComposer qui se trouve dans le dossier /Developer/Application/Graphics Tools/.

2. Prise en main

Au démarrage, QuartzComposer vous propose de créer une nouvelle composition à partir d'un modèle. Pour mieux comprendre le fonctionnement de ce programme nous allons créer une Basic Composition. Cette dernière permet de découvrir les deux fenêtres principales du programme. La première est l'éditeur dans lequel se trouve le graphe de la scène dans la vue principale, et la bibliothèque d'éléments à gauche. La seconde fenêtre est le moniteur, qui affiche en temps réel la scène présentée dans l'éditeur.

Les éléments d'une scène sont appelés des patches et peuvent être interconnectés entre eux. La composition montre par exemple une Interpolation reliée à un Sprite. Tous les patches disponibles se trouvent dans la bibliothèque à gauche et il suffit d'un glisser déposer pour les insérer dans la composition. Vous constaterez la présence d'un onglet "Clip" au dessus de la liste des patches. Les clips sont des compositions que vous pouvez réutiliser au sein d'autres compositions. Essayez par exemple d'ajouter le clip Rotating Cube à la scène et observez le changement immédiat dans le moniteur. Si vous observez le clip que vous venez d'intégrer à la scène, vous constaterez que sa représentation utilise des bords rectangulaires et non arrondis comme les autres patches. Ces patches spéciaux indiquent qu'ils contiennent d'autres patches. Vous pouvez naviguer parmi les sous patches en double-cliquant sur un agrégat de patches ou en cliquant sur le bouton Hierarchy Brower. Le bouton Edit Parent permet de revenir à tout instant au patch parent. Par défaut, toute composition est donc créé dans un patch racine que nous ne pouvons pas voir ni éditer.

Le moniteur est indispensable pour concevoir mais également pour déboguer votre composition. En affichant la barre d'outils vous découvrirez plusieurs boutons indispensables. Vous pouvez ainsi arrêter le rendu, ce qui est appréciable dans le cas de composition très gourmandes en temps processeur, passer en mode plein écran, modifier les paramètres de la scène lorsqu'ils existent ou encore activer les modes de rendu de débogage. Nous allons à présent créer notre propre composition : revenez dans l'éditeur et supprimez tous les patches.

3. Animation de texte

Pour concevoir une scène Quartz, nous pouvons utiliser trois sortes de patches, identifiables par leur couleur mais également par leurs ports. Les patches verts représentent des opérations, possédant des ports en entrée et en sortie. Les patches cyan sont des entrées et n'ont donc que des ports en sortie. Enfin, les patches magenta servent à l'affichage (à la sortie de manière générale) et n'ont que des ports en entrée. Pour relier un port à un autre il vous suffit de cliquer dans le cercle le représentant et de glisser votre souris, sans lâcher le bouton, jusqu'au port de destination. En procédant ainsi vous transmettez la valeur d'un port de sortie à un port d'entrée.
Nous allons commencer par un exemple simple d'affichage de texte que nous compliquerons progressivement. Déposez dans un premier temps un patch Image With String sur la scène. Vous pouvez atteindre un patch rapidement en tapant son nom dans le champ de recherche de la barre d'outils. Ce patch permet de générer une image à partir d'une chaîne de caractères et de plusieurs autres paramètres optionnels. Sélectionnez le patch puis ouvrez l'inspecteur à l'aide du bouton de la barre d'outils. L'inspecteur permet de modifier les paramètres et les arguments d'entrée d'un patch. Les paramètres de notre patch comportent par exemple la police de caractère. Les arguments d'entrée servent à attribuer des valeurs aux ports. Nous voulons ici changer la String pour afficher "Mostly Harmless". Le moniteur n'affiche malheureusement toujours rien. Vous devez en effet récupérer l'image produite par le patch et l'afficher sur une surface de dessin à l'aide d'un patch de rendu. Dans la bibliothèque, sélectionnez un Billboard et déposez-le sur la scène. Reliez ensuite le port Image du patch Image With String au port Image du Billboard pour que votre texte apparaisse enfin à l'écran.

Ajoutez à présent un gradient et éditez ses paramètres pour choisir des couleurs qui vous plaisent. Si vous consultez le moniteur vous constaterez que votre texte a disparu ! Pour y remédier, vous devez changer l'ordre des patches d'affichage. Chaque patch magenta dispose d'un numéro dans son coin supérieur droit qui indique dans quel ordre ils sont affichés. Puisque le gradient possède le numéro 2, il s'affiche par-dessus votre texte. Affichez le menu contextuel du gradient et choisissez le Rendering Layer 1. Le résultat pourrait être satisfaisant si le texte ne présentait pas un horrible fond noir. Chaque layer de rendu dispos d'un paramètre de mélange appelé Blending. Modifiez celui du Billboard et sélectionnez le mode Over.

Nous allons à présent modifier le texte pour qu'il s'affiche avec un effet d'illumination. Une méthode répandue consiste généralement à appliquer un effet de flou au texte puis à redessiner le texte par-dessus le flou. On obtient ainsi un subtil halo. Quartz Composer vous permet de faire cela en quelques clics. Déposez tout d'abord un patch Gaussian Blur sur la scène et utilisez la sortie Image de l'Image With String sur l'entrée Image. Vous pouvez vérifier le résultat en branchant la sortie du flou sur le Billboard ou en laissant votre souris quelques instants au dessus du port de sortie. L'étape suivante consiste à réafficher le texte d'origine par-dessus le flou que nous venons de créer. La solution naïve consiste à utiliser un second Image With String. Fort heureusement, nous pouvons utiliser une même sortie avec plusieurs entrées. Ajoutez donc un patch Source Over à la scène. Ce patch accepte deux images en entrée et les mélange pour créer une image unique en sortie. Utilisez l'Image With String comme argument Image et le Gaussian Blur comme argument Background Image. Il ne vous reste plus qu'à connecter le résultat au Billboard pour admirer l'effet.

Ce dernier serait encore plus impressionnant s'il était animé. Nous allons donc faire en sorte que le halo grandisse et diminue pour simuler des pulsations. La technique consiste à utiliser un générateur de valeur et à le brancher sur le port Radius du flou qui contrôle la diffusion de celui-ci. Quartz Composer propose un outil parfaitement adapté, le patch LFO (Low Frequency Oscillation). Ce patch génère des valeurs au cours du temps qui suivent une fonction mathématique paramétrable. Nous allons choisir une sinusoïde, qui permet des allers-retours adoucis, d'amplitude 5 et de période 4. L'amplitude détermine et l'offset servent à définir les valeurs minimales, offset - amplitude, et maximales, offset + amplitude, générées par la sinusoïde. La période définit le nombre de secondes nécessaires pour parcourir toutes les valeurs. Terminez en reliant le port Result du LFO au port Radius du Gaussian Blur.

L'idée semblait bonne mais le résultat est une catastrophe. Le texte change de taille et semble se comporter de manière erratique. En étudiant la description du Gaussian Blur on apprend cependant que cet effet modifie la taille de l'image d'origine. Nous devons donc modifier la scène de manière à avoir une image de taille fixe, quelle que soit le rayon du flou gaussien. La meilleure solution consiste à recadrer l'image avec le patch Crop. Ses arguments en entrée comprennent une image et un rectangle de recadrage. Les unités utilisées pour exprimer ce rectangle sont en pixel, or les dimensions de l'image fournies par Image With String sont en unité Quartz. Nous devons utiliser un nouveau patch, Image Dimensions, capable de convertir les unités Quartz en pixels. Les ports Pixel Wide et Pixel High doivent donc être reliés aux ports Rectangle Width et Rectangle Height du patch Crop. N'oubliez pas d'utiliser l'Image With String en entrée. Passez le moniteur au premier plan et admirez votre animation.

4. Diffusez vos compositions

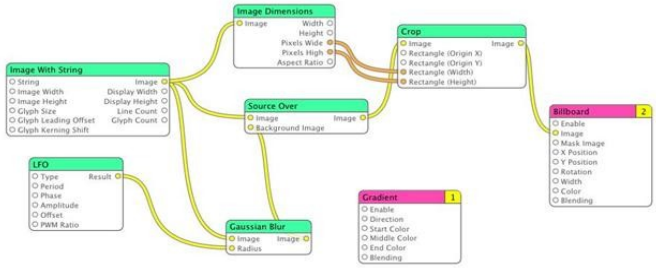
Quartz Composer vous propose plusieurs méthodes de diffusion de vos compositions. La plus simple est le format QTZ, reconnu par QuickTime 7 et Quartz Composer. Enregistrez simplement la composition pour obtenir un tel fichier. Sachez toutefois que la distribution d'une composition dans ce format revient à fournir le code source puisque vous pouvez l'ouvrir dans Quartz Composer pour étudier les patches, les liens et les paramètres. Nous vous encourageons naturellement à opter pour cette solution. Si vous désirez néanmoins protéger votre travail, vous pouvez exporte la scène au format QuickTime en appuyant sur Pomme-E. Vous pourrez alors choisir une résolution et une durée. Cette solution est extrêmement intéressante pour intégrer des effets, notamment pour des génériques, dans des films iMovie.

Vous pouvez également utiliser vos compositions comme économiseur d'écran. Il suffit de la copier dans le dossier /Library/Screen Savers ou dans le dossier ~/Library/Screen Savers pour la voir apparaître dans le dialogue de sélection de l'économiseur d'écran. Vous pouvez laisser à l'utilisateur la possibilité de le personnaliser en publiant des arguments d'entrée. Affichez le menu contextuel de l'un de vos patches et choisissez un argument dans Published Inputs. Vous pouvez par exemple publiez l'argument String du patch Image With String. Vous pouvez enfin réutiliser vos compositions sous forme de clips en éditant leur informations depuis le menu d'édition puis en les copiant dans le dossier /Library/Application Support/Apple/Quartz Composer/Clips.

N'hésitez pas à essayer les dizaines de patches proposés par Quartz Composer et vous découvrirez rapidement d'innombrables façons de réaliser d'impressionnantes démos graphiques.

5. Screenshots

Notre première composition Quartz Composer !



Les scènes se compliquent rapidement.

Retrouvez l'article de Romain Guy en entier en ligne : Lien50

Actualité WWDC07

WWDC 2007: Rien que Leopard !?

C'est cette semaine que le WWDC, le rendez-vous des développeurs sous Mac OS X et pour Mac OS X, a lieu.
Avec son lot d'annonces, de suprises, ...

Demain matin, en lisant ce blog, vous aurez droit à notre compte rendu du Keynote tenu par Steve Jobs. Pour les impatients, sachez qu'il sera publié ce soir aux environs de minuit (heure de Paris).

Ensuite, tout dépendra de ce qui filtrera jusqu'à nous, vu que les autres sessions ne sont pas publiques.

La vedette de cette édition 2007 de WWDC devrait être Mac OS X 10.5 (Leopard), puisqu'Apple devrait dévoiler les fameuses fonctionnalités qu'il a tenu secrètes jusqu'à présent. (Mais existent-elles vraiment ?)
On espère vraiment qu'Apple ne nous décevra pas sur ce point là. Et que ce que Steve annoncera sera vraiment quelque chose qu'il aura réussi à tenir secret jusqu'à présent. Car toutes les rumeurs qui nous sont parvenues jusqu'à présent n'ont rien de révolutionnaire, à part peut être une seule qui voudrait qu'on aurait plus besoin de parallels ou VMWare pour faire tourner les applications Windows sous Mac OS X Leopard.

Certains s'attendent à ce qu'Apple dévoile également les nouveaux iMacs. Et on aura peut être droit à la présentation d'un SDK pour l'iPhone.

Les développeurs Java qui attendent depuis de nombreux mois après la version définitive d'un JDK 6 pour Mac OS X ne devraient pas être déçus.

Le Keynote de Steve Jobs: un peu décevant

Voilà. C'est fait. Steve Jobs a fini son Keynote d'ouverture de l'édition 2007 du WWDC.

Après avoir débuté le Keynote avec un nouvel épisode de "I am a Mac, I am a PC", Steve Jobs est monté sur scène pour remercier les plus de 5000 personnes qui étaient présentes ce jour là. Et qui allaient pouvoir assister durant toute la semaine aux 159 sessions et 94 labos.

Il a ensuite remercié Intel pour le travail qu'ils ont accomplis ensemble pour migrer vers la plateforme Intel.

Ensuite, Steve a laissé la place à EA et Id Software qui ont tous les deux annoncés de nouveaux jeux sous Mac.

Steve est à nouveau réapparu pour nous parler de Mac OS X Leopard.
Tout d'abord, il a brossé rapidement la situation actuelle:

22Millions d'utilisateur Mac OS X dont
15 Millions pour Tiger,
5 Millions pour Panther,
2 Millions pour les versions précédentes

Steve annonce qu'il va nous parler de 10 nouvelles fonctionnalités, parmis les 300 fonctionnalités qui font leur apparition.
Malheureusement, aucune trace de nouvelles fonctionnalités tenues secrètes jusqu'à présent.
C'est tout de même une assez grosse déception.
Mais voyons malgré tout les 10 fonctionnalités dont il nous a parlé:

Nouveau Desktop
Rien de bien révolutionnaire: un peu de transparence, la barre de menu qui s'adapte à la photo en arrière-plan, et la notion de "stack", des dossiers que l'on peut rajouter au dock.

Nouveau Finder
avec une nouvelle barre latérale, la possibilité de rechercher d'autres Macs et Serveurs sur le réseau, partager les fichiers, ...

QuickLook
Permet de prévisualiser des fichiers sans devoir ouvrir l'application. Rien de vraiment novateur là dedans.

64 Bits
Non seulement l'OS est 64 Bits, mais Cocoa l'est également. De plus, l'OS permettra de faire tourner cote à cote des applications 32 bits et des applications 64 bits.
Pour montrer le gain de puissance que peut apporter le passage au 64 bits, il a montré une application écrite en 32 bits, et la même écrite en 64 bits. Cette application chargeait une image de 4Go. Il pouvait zoomer dessus, ... Tout cela, 2 à 3 fois plus rapidement avec l'application 64bits que l'application 32 bits.

Core Animation
Au programme, accélération matérielle, permet de faire des animations sans déployer de grands efforts de programmation. Support pour Textes, images, vidéos, OpenGL

Boot Camp
Directement intégré à Leopard. Plus besoin de graver un CD pour obtenir les drivers. Le compagnons idéal pour VMWare et Parallels

Spaces
Vous permet de regrouper vos applications dans des "espaces virtuels" et de passer d'une application à l'autre dans cet espace. Facile de réorganiser ses espaces, ...

DashBoard
Bien que DashBoard ne soit pas une nouveauté, Steve a voulu remercier les développeurs ici présent pour l'incroyable nombre de widgets disponibles pour DashBoard: plus de 3000 widgets.
Il a aussi montré combien il était simple de faire son propre Widget, grace à DashCode.

iChat
Enfin, support de la vidéo. Possibilité de faire des trucages en live, ...

Time Machine
La fonctionnalité la plus spectaculaire, je pense. Va grandement simplifier les backups automatiques.

Et voila que Steve a déjà fini de nous présenter les 10 fonctionnalités, parmis les 300 nouvelles fonctionnalités. Malheureusement, pas de trace d'une fonctionnalité tenue secrète. Ca rend la fête moins belle.

Steve a terminé par un petit clin d'oeil, annonçant que la version basique serait à 129$, tout comme la version premium et business. Et même la version ultimate sera à 129$. Rajoutant que tout le monde achetera très certainement la version Ultimate. A mourir de rire.

Il a tout de même gardé quelque chose pour la fin. Mais cela n'est pas pour nous, utilisateur Mac. Non. Le cadeau surprise de Steve est pour les utilisateurs Windows, puisque la version 3.0 de Safari est maintenant disponible (en version Béta) pour Windows. Quelques chiffres montraient que Safari est 2 fois plus rapide qu'Internet Explorer, 1.5 fois plus rapide que Mozilla FireFox.

Apple espère réussir avec Safari le même exploit qu'il a réussi avec iTunes. Et espère encore augmenter sa part de marcher des navigateurs Web, actuellement de 5%

Steve Jobs a finalement cloturé le KeyNote avec l'iPhone, indiquant qu'il sera possible de développer des applications pour l'iPhone. Que ces applications se développeront comme des pages web traditionnelles, Web 2, avec AJAX, ... mais pouvant appeler des fonctionnalités embarquées de l'iPhone. Ces applications seront déployées sur un serveur et tourneront dans un espace protégé (sandbox) sur l'iPhone.

Et voilà. C'est tout.

Conclusion
On est tout de même assez déçu. Pas d'annonces spectaculaires. Pas de fonctionnalités top secret dévoilées. On est resté quelque peu sur sa fin.

Mais, il y a quelques affiches qui sont encore drappées de noir. Que se cache-t-il là-dessous ? On le saura dans les prochaines heures.

Developpez sous Leopard: encore plus facile

WWDC étant avant tout un salon pour les développeurs, il est normal que les sessions soient orientées développement.

Il y en avait pour tout les goûts.

Le Matin
Cela allait des démos données aux administrateurs systèmes, pour qu'ils puissent réveiller les Mac grâce au "Wake on Lan" depuis une application Web écrite en Ruby (on Rails).

En passant par une présentation des services offerts par la version Serveur de Leopard (celle là n'est pas à 129€.)
Une autre session était consacrée aux améliorations apportées dans Leopard pour le développement d'applications qui doivent interagir avec des "Smart Card".

Pendant ce temps là, il était possible d'apprendre comment rendre disponible le contenu en "broadcast" à l'aide de la fonction "theatre" d'iChat, ou encore, pour les programmeurs au niveau du noyau, apprendre les arcanes du "Kernel" et comment le déboguer.

Steve avait encouragé lors de son Keynote à continuer à fournir des Widgets pour DashBoard. Une session était consacrée au développement de Widgets à l'aide de DashCode.

L'après-midi
Alors que le matin il y avait aussi des sessions pour Core Data, cet après midi, c'était Core Audio, Core Animation, Cocoa, "Bindings" sous Cocoa, "GUI Animée" en Cocoa, ...

Toujours également des sessions pour les administrateurs systèmes, mais aussi des sessions proches du noyau, comme savoir ce qui se passe entre le moment où le Mac démarre et le moment où vous devez vous identifier. Notez qu'au programme il était question d'expliquer comment "booter" depuis des systèmes de fichiers non reconnus par la ROM.

Le soir
Le soir, c'était la remise des trophées.

A noter, dans la catégorie Outils de Développement, CSSEdit 2.5 (Lien51), et dans la catégorie Projets Etudiants, Picturesque 1.0 (Lien52) pour traiter vos images en batches.

Developper sous Leopard ? Oui, mais sous MacIntel

Plus on avance dans la semaine, plus on se rend compte qu'Apple délaisse clairement de plus en plus la plateforme PPC.

Si vous êtes développeurs Mac ou sous Mac, vous êtes invité à passer aux MacIntel.

Qu'est ce qui nous fait dire cela ?

Leopard 64Bits
Tout d'abord, il faut savoir que lors du KeyNote de lundi, Steve Jobs avait présenté parmi les 10 nouveautés choisies sur les 300 contenues dans Leopard le fait que Leopard serait un OS 64 bits.

Seulement voilà. Il semble tout d'abord que Leopard ne sera pas disponible pour les Mac tournant avec un PPC G3. Leopard ne tournerait que sur un G4, G5 ou MacIntel.

Rosetta
Alors que Steve Jobs a bien dit que Cocoa serait également en 64 bits, il semblerait que Rosetta, qui permet de faire tourner des applications PPC sur Intel n'est pas 64 bits.
Un développeur Mac sous PPC est donc obligé de soit garder son application PPC en 32 bits pour qu'elle puisse également tourner sur les MacIntel via Rosette. Soit rendre son application 64bits disponible en UB (Universal Binary. Application ayant le code PPC et le code Intel. Plus d'émulation via Rosetta dans ce cas).

Java 6 uniquement sur MacIntel ?
Pour un développeur Java, ca risque d'être encore plus ennuyant.
Vu que les sessions consacrées à Java était concentrées sur la fin

de la semaine principalement, il nous a fallu attendre un peu plus pour savoir de quoi il retourne.

Mais, on peut déjà dire que nos dires se confirment.

Tout d'abord, Apple a profité du WWDC pour mettre à jour son site. Du moins la partie anglaise. Or on retrouve sur cette page consacrée au 64Bits (vu que Léopard sera un OS 64Bits) la mention suivante:

64-bit frameworks.

In addition to the POSIX and math libraries supported in Tiger, Leopard enables developers to build complete 64-bit applications using the Cocoa, Quartz, OpenGL, and X11 GUI frameworks. ***You can even use 64-bit Java on capable Intel processors****. And the 64-bit and 32-bit versions of the libraries are built from exactly the same code base, to ensure a consistent experience for both developers and users.*

(c'est nous qui soulignons)

De plus, quelqu'un qui était présent au WWDC a dit ceci concernant la session « Discover Java for Leopard ».

Pas vraiment de grosses nouvelles, sauf que si vous êtes intéressés par Java 6, vous êtes mieux de passer à Intel si vous êtes encore sur PPC.

Qu'est ce que cela signifie.

Si en s'en tient à ce que dit Apple,
a) il y aura bien un JVM 64Bits disponible pour léopard.
b) la version 64bits de la JVM sera disponible UNIQUEMENT pour ceux qui ont des Mac Intels. Ce qui est dommage pour les possesseurs de Mac PPC.

Si on recoupe cela avec les déclarations de la personne présente au WWDC,
c) Il est évident que cette version 64Bits de Java sera Java 6.

Mais rien ne dit qu'il y aura une version 32 bits de Java 6 disponible pour les Mac PPC. Assez génant, n'est-ce pas. (remarquez que rien ne dit le contraire non plus. Faudra attendre la verion public de Leopard pour savoir finalement quoi)

Les jeux uniquement sur MacIntel
Toujours durant le Keynote, Steve Jobs a fait monter sur scène des patrons de société de jeux annoncant que les possesseurs de Mac pourraient jouer. Seulement voilà. Le même jour, TransGaming fait savoir à tous que cela est rendu possible grâce à la technologie Cider qu'elle vend. Le seul hic. C'est que cela n'est possible que sous les MacIntel. Les possesseurs de Mac PPC ne pourront pas profiter de ces jeux. Et vous, en tant que développeur, il vous faut développer vos jeux sous Windows, et utiliser Cider pour les faire tourner sous Mac OS X mais uniquement sur des machines MacIntel.

Conclusion
Il est clair que les Mac PPC ont vraiment un pied dans la tombe. Et qu'un développeur Mac a intérêt, si cela n'est pas encore fait, à s'acheter un MacIntel rapidement après la sortie de Leopard. Pour les développeurs Java, la question est de savoir si Apple sortira tout de même une version 32 bits de Java SE 6 pour les Mac PPC. C'est à espérer.

Java 5 ou Java 6 sous Leopard ? La saga continue

Nous vous annoncions fin du mois passé que Java serait disponible en 64bits sur Leopard.
Hier, nous vous avions dit que cela ne concernerait que les MacIntel, et que ce serait bien une implémentation de Java6.

Mais voilà. nous venons de tomber sur une autre page provenant du site d'Apple, qui contredit le fait que ce serait une implémentation de Java 6.

Lorsqu'on se rend sur les pages concernant la version Serveur de Mac OS X Leopard, on peut lire sur celle ci les 2 informations suivantes:

Many core services — such as Apache 2, MySQL 5, Postfix, Cyrus, Podcast Producer, QuickTime Streaming Server 6, and ***Java VM on Intel*** *— are now 64-bit*

Ceci confirme donc bien que la JVM sera en 64 Bits.
Par contre, sur la même page, mais un peu plus bas, il est dit ceci

---

If offers administration of either Apache 2.2 or 1.3, MySQL 5 with Apache/MySQL/PHP integration, Tomcat 5, and ***a 64-bit Java 1.5 VM on Intel*** *for hosting enterprise applications*

La version Serveur de Leopard serait donc livrée avec une version 64 bits de Java 1.5. Et non Java 6.

Est-ce une erreur sur cette page ? En tout cas, elle contredit les informations publiées hier.
Qu'en est il exactement finalement ? On continue de creuser pour vous tenir informé.

Retrouvez toute l'actualité Mac sur le blog Mac : Lien53

Les derniers tutoriels et articles

Ajax facile avec Ruby on Rails, Prototype, script.aculo.us et les RJS

Nous allons voir au cours de cet article la facilité de mise en place qu'offre Rails pour faire de l'ajax, les superbes effets que cela permet, et tout cela sans une ligne de Javascript ou presque, grâce aux librairies Prototype et script.aculo.us.
Nous verrons également l'intérêt des RJS.

1. Introduction

AJAX est une technologie extrêmement intéressante à mettre en place, puisqu'elle permet une interaction forte avec le client. L'idée est de pouvoir mettre à jour des éléments de la page sans avoir à recharger la page complète.
Qui dit AJAX dit Javascript, mais Rails va nous permettre de faire tout cela sans en faire une ligne, ou presque.

Nous commencerons par un petit rappel pour les moins à l'aise d'entre vous, les autres peuvent passer directement au chapitre III (si vous êtes débutant avec Rails, vous pouvez aller consulter le tutorial "Initiation à Rails" (Lien54), par Yann Marec).

Dans ce cours, nous allons utiliser les différentes librairies Javascript de Rails.
Afin que ces librairies soient déclarées dans toutes nos pages, nous allons définir un fichier "app/views/layouts/application.rhtml" comme suit :

application.rhtml
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <%= javascript_include_tag :defaults %>
        <title><%= @body_title || "titre par défaut"
%></title>
    </head>
    <body<%= @body_extras %>>
        <%= yield %>
    </body>
</html>
```

Ce fichier se charge d'inclure les librairies Javascript dont nous aurons besoin, et nous permettra de stipuler pour chaque vue une valeur pour la balise **title**, et éventuellement des options à passer à la balise **body**.

2. Rappels

2.1. Génération d'un contrôleur

Pour générer un contrôleur, nous taperons la ligne suivante à la racine de notre application Rails : "*ruby script/generate controller nom_du_contrôleur*".
Par exemple, nous souhaitons créer un modèle "test", ce qui nous donnera : "*ruby script/generate controller test*". Voici ce que Rails vous renverra :

Retour lors de la création d'un contrôleur
```
exists  app/controllers/
exists  app/helpers/
create  app/views/test
exists  test/functional/
create  app/controllers/test_controller.rb
create  test/functional/test_controller_test.rb
create  app/helpers/test_helper.rb
```

Ruby nous a donc créé :
- **create app/controllers/test_controller.rb** : le contrôleur lui-même
- **create app/views/test** : le répertoire dans lequel nous placeront les vues associées au contrôleur
- **create test/functional/test_controller_test.rb** : un fichier dans lequel nous écrirons nos tests fonctionnels
- **create app/helpers/test_helper.rb** : un fichier qui contiendra les helpers (assistants) de notre contrôleur

2.2. Fichier de configuration de la base de données

Le fichier qui sert à configurer votre base de données est le suivant : "*config/database.yml*".
Il vous permet de configurer quelle base de données utiliser dans les trois modes de ruby : développement, test et production.

Voici un exemple pour le mode développement :

Contenu du fichier 'database.yml' pour le mode développement
development:
```
  adapter: mysql                    # Le type de la
base
  database: mon_application         # Le nom de la
base
  username: root                    # L'identifiant à
utiliser pour se connecter
  password:                         # Le mot de passe
  host: localhost                   # L'adresse du
serveur qui héberge la base
```

Configurez ce fichier en fonction de votre environnement de travail, et passons à la suite.

Il faut tout de même créer la base de données à la main.

2.3. Génération d'un modèle

Pour générer un modèle, nous taperons la ligne suivante à la racine de notre application Rails : "*ruby script/generate model nom_du_modele_au_singulier*".
Par exemple, nous souhaitons créer un modèle "utilisateur", ce qui nous donnera : "*ruby script/generate model utilisateur*". Voici ce que Rails vous renverra :

Retour lors de la création d'un modèle
```
exists  app/models/
```

```
exists  test/unit/
exists  test/fixtures/
create  app/models/utilisateur.rb
create  test/unit/utilisateur_test.rb
create  test/fixtures/utilisateurs.yml
create  db/migrate
create  db/migrate/001_create_utilisateurs.rb
```

Ruby nous a donc créé :

- **create app/models/utilisateur.rb** : le modèle lui-même
- **create test/unit/utilisateur_test.rb** : un fichier dans lequel nous écrirons nos tests unitaires
- **create test/fixtures/utilisateurs.yml** : un fichier qui nous servira à remplir notre base lors des tests (je l'utiliserai pour remplir la base avec un jeu de données initial)
- **create db/migrate/001_create_utilisateurs.rb** : un fichier de migration, qui va nous servir à créer la table au départ

2.4. Utilisation des fichiers de migration

Les fichiers de migration sont situés dans le répertoire "db/migrate/" de votre application.
Ils peuvent servir à créer la structure de vos tables, à la modifier ou à les supprimer.

Les fichiers de migration sont extrêmement puissants, puisqu'ils permettent via une simple ligne de commande de migrer entre les différentes versions de vos bases de données, tout en gardant une traçabilité totale, et ce quel que soit le type de base de données que vous utilisez.

Voici le fichier de migration généré en même temps que le modèle :

fichier de migration '001_create_utilisateurs.rb'

```
class CreateUtilisateurs < ActiveRecord::Migration
  def self.up
    create_table :utilisateurs do |t|
    end
  end

  def self.down
    drop_table :utilisateurs
  end
end
```

Supposons maintenant que nous souhaitions définir un nom et un âge pour un utilisateur.
Modifions le fichier comme suit :

fichier de migration '001_create_utilisateurs.rb'

```
class CreateUtilisateurs < ActiveRecord::Migration
  def self.up
    create_table :utilisateurs do |t|
      t.column "nom", :string, :null => false
      t.column "age", :integer, :null => true
    end
  end

  def self.down
    drop_table :utilisateurs
  end
end
```

Il ne nous reste plus qu'à exécuter ce fichier de migration pour avoir notre table crée à l'aide de la commande suivante : "rake db:migrate VERSION=1" (le numéro de version correspond aux trois premiers chiffres de votre fichier de migration).

```
(in mon/path)
== CreateUtilisateurs: migrating
==============================================
-- create_table(:utilisateurs)
   -> 0.1130s
== CreateUtilisateurs: migrated (0.1140s)
=====================================
```

Si vous allez voir la structure de votre table récemment créée, vous vous apercevrez que Rails s'est permis de rajouter un champ "id" auto-incrémenté. Ceci est parfaitement normal, Rails a besoin d'un champ nommé id auto-incrémenté dans toutes les tables (en respectant les conventions).

Il ne nous reste plus qu'à remplir notre base avec des valeurs par défaut.

2.5. Utilisation des fixtures

Il existe deux types de fichiers pour remplir votre base. Par défaut, rails vous a créé un fichier yml. Il est également possible de le renommer en csv.
Nous allons voir comment remplir ces deux types de fichier :

2.5.1. utilisateurs.yml

Voici le fichier tel que je l'ai modifié pour remplir ma base avec trois utilisateurs :

utilisateurs.yml

```
utilisateur1:  # Un identifiant quelconque pour
l'enregistrement en cours
  nom: Riri    # La valeur du champ "nom"
  age: 12      # La valeur du champ "age"

utilisateur2:
  nom: Fifi
  age: 13

utilisateur3:
  nom: Loulou
  age: 14
```

Ici, je n'ai pas pris la peine de préciser les id. Il peut tout de fois être judicieux de le faire, surtout au cours de tests où vous souhaitez connaître l'id d'un utilisateur précis (cas de jointures par exemple).

Si vous donnez deux fois le même identifiant à deux enregistrements différents, Rails remplacera le premier par le second.

2.5.2. utilisateurs.csv

Voici cette fois un fichier faisant exactement la même chose, mais au format csv :

utilisateurs.csv

```
nom, age
Riri, 12
Fifi, 13
Loulou, 14
```

Une fois de plus, je n'ai pas précisé les id.
La première ligne reprend simplement le nom des champs à remplir. Les lignes d'après contiennent les enregistrements, en suivant le "modèle" définit sur la première ligne.

2.5.3. Déploiement des fixtures

Il ne nous reste plus qu'à exécuter ce fichier pour remplir notre table.
Tapez, toujours à la racine de notre application, la ligne de commande suivante : "*rake db:fixtures:load FIXTURES=utilisateurs*".

Ruby ne vous affichera rien, mais les données sont bien insérées.

Il est possible d'exécuter plusieurs fixtures en même temps. Pour cela, tapez "*rake db:fixtures:load FIXTURES=nom_fixture1,nom_fixture2,...*".
Cependant, vous ne devez pas avoir d'espaces ni à droite ni à gauche de la virgule, sous peine d'erreur.

3. Les RJS (Remote JavaScript)

Les RJS sont des modèles Javascript destinés à mettre dynamiquement à jour une page.
Ils se placent dans le répertoire "views" de notre application (dans le sous-répertoire du contrôleur correspondant), et prennent simplement l'extension RJS.

Il faut considérer les RJS comme une vue qui met à jour les éléments de la page.

Une action du contrôleur et un RJS peuvent (et doivent dans le cas où le RJS est spécifique à une action) porter exactement le même nom. Dans ce cas, ils s'exécuteront l'un après l'autre, d'abord l'action puis le RJS.
Le fait de passer par le contrôleur peut permettre d'affiner les actions qui seront exécutées dans le RJS.

Voyons tout de suite un petit exemple d'utilisation d'un RJS (ne vous attardez pas sur le code, nous y reviendrons à la fin).
Je me suis basé sur le modèle et le contrôleur créés au chapitre II.

test_controller.rb
```
class TestController < ApplicationController
  def index
    @users = Utilisateur.find(:all)
  end
  def make_bigger
    @users = Utilisateur.find(:all,
:conditions=>"age=14")
  end
end
```

index.rhtml
```
<% @body_title = "Premier exemple de RJS"%>
<% @users.each do |user| %>
    <span id="element_<%= user.id %>"><%= user.nom
%></span>
    <br />
<% end %>
<hr />
<%= link_to_remote "Faire grossir les items (age =
14)",
            :url=>{:action=>"make_bigger"},
            :before => "Element.show('wait_icon')",
            :complete => "Element.hide('wait_icon')"
%>
<br />
<%= image_tag "ajax-loader.gif", :id=>"wait_icon",
:style=>"display:none" %>
```

Notez simplement la présence de la variable "@body_title" définie dans notre vue qui va nous permettre d'affecter à la vue en cours un titre.

Vous pouvez télécharger l'image de chargement ici.
Placez-la dans le répertoire "public/images/" de votre application.

make_bigger.rjs
```
@users.each { |user|
  page.call "new Effect.Scale", "element_#{user.id}",
200
}
```

Vous n'avez plus qu'à lancer votre serveur, à cliquer sur le lien et à admirer le résultat.

J'ai souhaité définir une action "make_bigger" dans mon contrôleur afin de récupérer la liste de mes utilisateurs avant de la passer au RJS du même nom (d'ailleurs, je ne l'appelle nulle part, Rails va le chercher tout seul).
Dans le cas contraire, j'aurais été obligé de répéter trois fois le même code dans mon RJS, en changeant l'identifiant (ou de faire un compteur, mais imaginez que mes id ne se suivent pas...).

Attention : un appel à un RJS est un appel serveur. Qui dit appel serveur dit temps de latence, utilisez-les donc en connaissance de cause.

Dans l'état actuel des choses, si la personne a désactivé Javascript, rien ne ne produira lors du click sur le lien. Pour comprendre pourquoi, il suffit de regarder le code source généré :

```
<a href="#" onclick="Element.show('wait_icon'); new
Ajax.Request('/test/make_bigger', {asynchronous:true,
evalScripts:true,
onComplete:function(request){Element.hide('wait_icon')}
}); return false;">Faire grossir les items</a>
<a href="/test/make_bigger">coucou</a>
```

Le href ne pointant sur rien, rien en se passera dans le cas où javascript est désactivé.
Mais il est possible de gérer ce problème.
Remplacez votre "link_to_remote" par le code suivant :

```
<%= link_to_remote "Faire grossir les items (age =
14)",
            {
                :url=>{:action=>"make_bigger"},
                :before => "Element.show('wait_icon')",
                :complete =>
"Element.hide('wait_icon')"
            },
            :href=>(url_for :action=>"make_bigger")
%>
```

Ici, nous lui précisons quelle valeur mettre dans l'attribut href de notre lien.
Il ne nous reste plus qu'à gérer dans notre contrôleur le cas ou la requête est une requête AJAX, ou bien une simple requête HTML.
Modifiez donc votre action "make_bigger" comme suit :

```
def make_bigger
       respond_to do |requete|
              requete.html {redirect_to
:action=>"index"}
```

```
                requete.js {
                        @users = Utilisateur.find(:all,
:conditions=>"age=14")
                        render :action=>"make_bigger.rjs"
                }
        end
end
```

Nous gérons désormais dans notre contrôleur les différentes actions à effectuer en fonction du type de requête.
Pour en savoir plus sur comment gérer le cas où Javascript est désactivé, regardez du coté du plugin UJS (Unobtrusive JavaScript).

4. Quelques petits effets graphiques

Nous allons voir dans ce chapitre quelques effets graphiques forts sympathiques, que nous apporte la librairie script.aculo.us.

4.1. Effets combinés

4.1.1. Fade / Appear / ToggleAppear

Disparition par fondu, apparition par fondu, et alternance entre les deux.

Créons un nouveau contrôleur nommé "effets", puis créez le fichier "index.rhtml" comme suit :

Fade / Appear / ToggleAppear
```
<%= link_to_function("Fade") do |page|
        page.visual_effect(:fade, "bout_de_page")
    end
%>
<%= link_to_function("Appear") do |page|
        page.visual_effect(:appear, "bout_de_page")
    end
%>
<%= link_to_function("ToggleAppear") do |page|
        page.visual_effect(:toggle_appear,
"bout_de_page")
    end
%>
<br /><br />
<div id="bout_de_page" style="background-color:#1188FF;
width:350px; height:150px;">
    <div>
        <%= image_tag "rails.png"%>
        <p>Bonjour à tous, ceci n'est qu'un petit
exemple des possibilités de <b>script.aculo.us</b></p>
    </div>
</div>
```

Je sais, les styles définis directement dans le code, on a déjà vu mieux, mais pour l'exemple ce sera amplement suffisant ;)

Il ne vous reste plus qu'à cliquer et à admirer.
Bien entendu, pour faire apparaitre un élément, il faut déjà l'avoir fait disparaitre. Commencez-donc par cliquer sur "Fade" avant de cliquer sur "Appear".

Pour tous les exemples suivants, nous contenterons d'ajouter de nouveaux liens nous le dernier "link_to_function" trouvé.

Le div imbriqué dans le premier n'est là que pour que certains effets passent parfaitement.

Juste à titre d'information, il est également possible d'écrire un "link_to_function" comme suit (reprenons l'exemple du ToggleAppear) :

```
<%= link_to_function "Hello (local)", update_page { |
page|
        page.visual_effect(:toggle_appear,
"bout_de_page")
    }
%>
```

Ce n'est qu'une question de choix.

4.1.2. Highlight

Mise en valeur visuelle d'un élément.

Highlight
```
<%= link_to_function("Highlight") do |page|
        page.visual_effect(:highlight, "bout_de_page")
    end
%>
```

4.1.3. SlideUp / SlideDown / ToggleSlide

Disparition vers le haut, apparition vers le bas, et alternance entre les deux.

SlideUp / SlideDown / ToggleSlide
```
<%= link_to_function("SlideUp") do |page|
        page.visual_effect(:slide_up, "bout_de_page")
    end
%>
<%= link_to_function("SlideDown") do |page|
        page.visual_effect(:slide_down, "bout_de_page")
    end
%>
<%= link_to_function("ToggleSlide") do |page|
        page.visual_effect(:toggle_slide,
"bout_de_page")
    end
%>
```

Notez simplement ici la séparation des deux mots qui s'est transformée en underscore (valable à chaque fois que vous rencontrerez le cas).

4.1.4. Grow / Shrink

Apparition en zoom, disparition en zoom.
Grow / Shrink

```
<%= link_to_function("Grow") do |page|
        page.visual_effect(:grow, "bout_de_page")
    end
%>
<%= link_to_function("Shrink") do |page|
        page.visual_effect(:shrink, "bout_de_page")
    end
%>
```

4.1.5. BlindUp / BlindDown / ToggleBlind

Disparition vers le haut, apparition vers le bas, et alternance entre les deux.

BlindUp / BlindDown / ToggleBlind
```
<%= link_to_function("BlindUp") do |page|
        page.visual_effect(:blind_up, "bout_de_page")
    end
```

```
%>
<%= link_to_function("BlindDown") do |page|
        page.visual_effect(:blind_down, "bout_de_page")
    end
%>
<%= link_to_function("ToggleBlind") do |page|
        page.visual_effect(:toggle_blind,
"bout_de_page")
    end
%>
```

4.1.6. Pulsate

Faire clignoter un élément.

Pulsate

```
<%= link_to_function("Pulsate") do |page|
        page.visual_effect(:pulsate, "bout_de_page")
    end
%>
```

4.1.7. Fold

Disparition vers le haut puis vers la gauche.

Fold

```
<%= link_to_function("Fold") do |page|
        page.visual_effect(:fold, "bout_de_page")
    end
%>
```

4.1.8. Puff

Disparition par zoom.

Puff

```
<%= link_to_function("Puff") do |page|
        page.visual_effect(:puff, "bout_de_page")
    end
%>
```

4.1.9. Squish

Disparition vers le haut et vers la gauche.

Squish

```
<%= link_to_function("Squish") do |page|
        page.visual_effect(:squish, "bout_de_page")
    end
%>
```

4.1.10. SwitchOff

Disparition à la manière de l'extinction des vieux téléviseurs.

SwitchOff

```
<%= link_to_function("SwitchOff") do |page|
        page.visual_effect(:switch_off, "bout_de_page")
    end
%>
```

4.1.11. DropOut

Disparition vers le bas.

DropOut

```
<%= link_to_function("DropOut") do |page|
        page.visual_effect(:drop_out, "bout_de_page")
    end
%>
```

4.2. Effets 'de base'

Tout ces effets sont très agréables, mais il est des moments où l'on aimerait se contenter de quelque chose de simple, ou tout simplement qui n'est pas proposé dans ces effets.
Voici comment en réaliser certains :

4.2.1. Insérer / Remplacer / Supprimer un élément

insert_html / update_html / remove

```
<%= link_to_function("Insérer") do |page|
        page.insert_html :bottom, "bout_de_page",
"coucou<br />"
    end
%>
<%= link_to_function("Remplacer") do |page|
        page.replace_html "bout_de_page", "<p>Le
contenu a été remplacé entièrement par du HTML et une
image</p>" + image_tag("rails.png")
    end
%>
<%= link_to_function("Supprimer") do |page|
        page.remove "bout_de_page"
    end
%>
```

Pour l'insertion, quatre paramètres sont disponibles, et vont définir l'endroit de l'ajout :

- **:top** : ajoute le paramètre dans la balise, tout au début
- **:bottom** : ajoute le paramètre dans la balise, tout à la fin
- **:before** : ajoute le paramètre immédiatement avant la balise
- **:after** : ajoute le paramètre immédiatement après la balise

Pour l'insertion et le remplacement de contenu, il est également possible de passer un partial en paramètre :

```
<%= link_to_function("Remplacer") do |page|
        page.replace_html "bout_de_page", :partial =>
'autre_contenu'
    end
%>
```

Il vous suffit de créer le partial "*_autre_contenu.rhtml*".
Vous pouvez également passer un objet au partial via le paramètre ":object".

Attention : Si vous regardez le code source généré, vous remarquerez que le partial est "embarqué" dans le code JavaScript (puisqu'aucun appel serveur n'est fait).
Cela peut donc considérablement alourdir le poids de vos pages.

De plus, imaginez que vous faisiez ainsi appel à un partial, qui lui-même à une fonction de ce type pour rappeler le premier partial (par exemple pour changer de formulaire). Et bien vous venez de créer une jolie boucle infinie qui se fera un plaisir de faire exploser votre serveur web. Prudence donc !

4.2.2. Cacher / Afficher un élément / alterner entre les deux

show / hide / toggle

```
<%= link_to_function("Show") do |page|
        page.show "bout_de_page"
    end
%>
<%= link_to_function("Hide") do |page|
        page.hide "bout_de_page"
```

```
    end
%>
<%= link_to_function("Toggle") do |page|
        page.toggle "bout_de_page"
    end
%>
```

4.2.3. Agrandir / Réduire un élément

Scale

```
<%= link_to_function("Scale 200%") do |page|
        page.call "new Effect.Scale", "bout_de_page",
200
    end
%>
<%= link_to_function("Scale 50%") do |page|
        page.call "new Effect.Scale", "bout_de_page",
50
    end
%>
```

4.2.4. Modifier le style d'un élément

Morph

```
<%= link_to_function("Morph bg-color") do |page|
        page.call "new Effect.Morph", "bout_de_page",
:style=>'background-color:#CC55DD;'
    end
%>
```

4.3.Récapitulatif

Dans tout ce chapitre, nous n'avons pas fait une ligne d'AJAX, l'intérêt étant inexistant en l'état. En effet, qui dit appel AJAX dit appel serveur, totalement inutile pour simplement changer l'aspect d'un élément.
Mais notez que ces effets pourront être utilisés exactement de la même façon depuis un RJS (page.XXX).

Tous ces effets vous serviront lorsque nous attaquerons AJAX, pour agrémenter de jolis effets vos transitions.

Maintenant que vous avez tout ces beaux effets dans vos bagages, vous devriez être capable d'imaginer les combinaisons les plus folles.
Voyons maintenant comment modifier dynamiquement des formulaires.

Retrouvez la suite de l'article de Pierre-Baptiste Naigeon en ligne : Lien55

Les livres Ruby

Ruby par l'exemple

Ruby connaît une popularité grandissante suite à l'engouement récent pour Ruby on Rails. Non content d'être un langage objet interprété complet et puissant, Ruby concentre nombre des qualités propres à satisfaire les développeurs les plus tatillons.

Ruby par l'exemple se propose de vous faire entrer dans l'univers de Ruby sans repasser par les bases du langage. Il s'adresse aux développeurs issus d'horizons aussi variés que celui des scripts en Perl ou Python, de la programmation objet en Java ou du Web avec PHP. Grâce à sa syntaxe à la fois simple et intuitive, Ruby permet de réaliser très rapidement des opérations qui seraient soit longues à mettre en place sous un autre langage, soit complexes à coder. Les 432 recettes de cet ouvrage sont là pour vous faire accéder directement à la solution adaptée au problème que vous avez à résoudre.

Critique du livre par la rédaction (Frédéric Jay)

Nombre de philosophes vous diront qu'il y a la théorie d'abord et la pratique ensuite.
Apprendre par la pratique, c'est ce que vous propose ce livre.

On dit aussi qu'un petit dessin vaut mieux qu'un long discours. Le livre applique à la lettre cette philosophie.
Le livre regorge d'exemples, souvent simples, courts et efficaces. De plus, la quantité des exemples couvre un large éventail, et donne plein de bonnes idées. Vous êtes sûrs de trouver des choses qui vous serviront un jour.
Si vous êtes débutant, vous serez épatés, en le feuilletant, de toutes les possibilités de ce langage.

Au niveau ergonomie, l'organisation du livre est assez bonne, on trouve facilement ce qu'on cherche; rien n'à envier de notre moteur de recherche préféré...
Et avoir le livre sous la main permet de noter ses idées et commentaires au fur et à mesure.

Attention tout de même, ici seul Ruby est traité.
Si vous voulez faire du Ruby on Rails, ce livre ne correspondra pas à votre attente dans un premier temps. Il vous servira par la suite pour faire des scripts en Ruby sur les serveurs.

En résumé, ce livre est extrêmement bienvenu, lorsque vous avez besoin de voir un bon exemple qui marche, pour vous aider.

Retrouvez ce livre sur la rubrique Jeux : Lien56

Les derniers tutoriels et articles

Introduction au développement en couches

Lorsque l'on est débutant en programmation, on entend souvent dire qu'il est important de développer ses applications en utilisant des couches, en séparant le code logique de l'interface utilisateur, etc.… Mais pour un débutant, toutes ces phrases sont abstraites et ne signifient rien. Au travers de cet article, nous allons voir, avec des exemples simples et concrets, comment réaliser une application utilisant ces différentes couches. Nous apprendrons leur importance, ce qu'elles représentent et comment les utiliser pour développer une application.

1. Introduction

Développer une application, tout le monde (ou presque) est capable de le faire : il suffit de concevoir l'interface utilisateur, puis de rajouter le code qui va bien sur le clic d'un bouton par exemple.

Si l'on est débutant, le premier réflexe que l'on va avoir sera d'insérer le code d'accès aux données (listes des clients, des produits, etc.…) directement sur le clic du bouton. Cette technique, bien qu'entièrement fonctionnelle, n'est pas idéale. En effet, imaginez que l'on vous impose que la liste des clients ne proviennent pas d'une base de données mais d'un fichier XML. Vous seriez alors obligé de modifier tout le code de votre application avant de la redéployer sur des postes clients. Si à l'inverse vous aviez utilisé le développement en couche, vous n'auriez qu'à modifier la couche d'accès à la liste de clients. Une application « en couche » est généralement composée d'au moins 4 couches :

- **L'interface graphique** (GUI, Graphic User Interface)
- **Les objets métier** (BusinnessObjects)
- **La couche d'accès aux données** (DAL, Data Access Layer)
- **La couche métier** (BLL, Business Logic Layer)

Personnellement, j'ai également l'habitude de rajouter une couche supplémentaire (Tools) qui me sert à regrouper diverses classes communes aux couches (par exemple les exceptions personnalisées).

Nous allons donc détailler chacune de ces couches et je vais tâcher de vous montrer comment les implémenter. Attention, il s'agit ici des méthodes de travail que j'utilise : rien ne garantit qu'elles soient optimales car le but est de vous démontrer comment essayer de bien développer votre application.

Afin de faciliter la compréhension de chacun, nous allons partir du principe que nous développons une simple application nous permettant de lister l'ensemble des clients d'un magasin (ces informations proviennent de la base de données).

2. Les couches

2.1. L'interface graphique

Pour notre interface graphique, nous allons faire quelque chose de simple : une fenêtre avec une ListBox (pour afficher la liste des clients), une TextBox et 2 boutons. En ce qui concerne la technologie utilisée, j'ai choisi WPF mais tout ce qui sera dit au cours de cet article s'applique également à du WindowsForms classique et à de l'ASP.NET. Voici le code utilisé :

```
<Window x:Class="DeveloppementNTiers.GUI.Window1"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/pre
sentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="DeveloppementNTiers.GUI" Height="300"
Width="300" Loaded="WindowLoaded" >
<StackPanel Orientation="Vertical">
<Button x:Name="btGetClients" Content="Get Clients"
Click="btGetClientsClick" />
<ListBox x:Name="lbClients" Margin="5"
ItemsSource="{Binding}" />
<TextBox x:Name="tbClientName" Margin="5"
Text="{Binding ElementName=lbClients,
Path=SelectedItem.ClientLastName}" /
<Button x:Name="btUpdateClientName" Content="Update
Client Name" Click="btUpdateClientNameClick" Margin="5"
/> </StackPanel>
</Window>
```

Il vous faut aussi rajouter, à votre application, un fichier de configuration dans lequel vous allez indiquer la chaîne de connexion à la base de données :

```
<?xml version="1.0" encoding="utf-8" ?> <configuration>
<connectionStrings> <add name="DBConnectionString"
connectionString="Data Source=T-
THOLE05\SQLEXPRESS;Initial
Catalog=AdventureWorks;Integrated Security=True"
providerName="System.Data.SqlClient" />
</connectionStrings> </configuration>
```

2.2. Les objets métier (Business Objects)

Les objets métier correspondent à tous les objets spécifiques que vous allez manipuler. Typiquement, dans notre cas, nous allons manipuler des clients. Un client, c'est un identifiant, un nom, un prénom, etc.… Toutes ces caractéristiques pourraient très bien représenter les propriétés de notre objet Client :

```
public class Client {
        private int m_ClientID;

        public int ClientID {
                get { return m_ClientID; }
                set { m_ClientID = value; }
        }

        private string m_ClientLastName;

        public string ClientLastName {
                get { return m_ClientLastName; }
                set { m_ClientLastName = value; }
        }
```

```
        private string m_ClientFirstName;

        public string ClientFirstName {
                get { return m_ClientFirstName; }
                set { m_ClientFirstName = value; }
        }

        public Client(int id, string lastname, string
firstname) {
                this.ClientID = id;
                this.ClientLastName = lastname;
                this.ClientFirstName = firstname;
        }
}
```

Si vous aviez eu besoin de manipuler des produits, vous auriez très certainement créé une classe Product, etc..

Dans notre cas, nous avons-nous-même écrit le code de cette classe. Généralement, on part du principe qu'une table dans la base de données représente un objet. Et que chaque colonne de la table représente une propriété de l'objet. Si vous aviez eu 50 tables dans votre base de données, vous auriez pu avoir besoin de 40 objets (certaines tables ne représentent pas forcément un objet, comme par exemple les tables temporaires, les tables système, etc.…). Pour générer le code correspondant (et non pas l'écrire à la main), vous pouvez utiliser ce que l'on appelle le « Mapping Objet-Relationnel », mais il s'agit là d'un autre sujet.

2.3. La couche d'accès aux données (DAL, Data Access Layer)

C'est dans cette partie que vous allez gérer tout ce qui concerne l'accès aux données. Il y a, dans cette couche, plusieurs parties bien distinctes :

- Une partie pour tout ce qui concerne l'accès à la base de données
- Une partie pour tout ce qui concerne les requêtes sur la base de données

Pour la partie « accès à la base de données », nous allons créer une classe qui se chargera de la connexion :

```
public class Sql {
        private DbConnection m_SqlCnx = null;
        private static Sql s_Instance;
        private static object s_InstanceLocker = new
object();
        private ConnectionStringSettings
m_CnxStringSettings;

        public ConnectionStringSettings
CnxStringSettings {
                get { return m_CnxStringSettings; }
                set { m_CnxStringSettings = value; }
        }

        // Singleton
        public static Sql Instance {
                get {
                        lock (s_InstanceLocker) {
                                if (s_Instance == null) {
                                        s_Instance = new
Sql();
                                }
                                return s_Instance;
                        }
                }
        }

        //Cette méthode créé une connexion à la BDD et
la renvoit
        public DbConnection GetSqlConnection() {
                DbProviderFactory factory =
DbProviderFactories.GetFactory(CnxStringSettings.Provid
erName);

                if (m_SqlCnx == null) {
                        m_SqlCnx =
factory.CreateConnection();
                }

                m_SqlCnx.ConnectionString =
CnxStringSettings.ConnectionString;

                if (m_SqlCnx.State ==
System.Data.ConnectionState.Closed) {
                        m_SqlCnx.Open(); } return
m_SqlCnx;
                }

        public void CloseConnection() {
                if (this.m_SqlCnx != null) {
                        if (this.m_SqlCnx.State !=
System.Data.ConnectionState.Closed) {
                                this.m_SqlCnx.Close();
                        }
                }
        }
}
```

Cette classe utilise un « design pattern » que l'on appelle « Singleton » : il s'agit d'une technique utilisée pour limiter l'instanciation d'une classe à un seul objet. Il nous faut à présent écrire la classe qui va exécuter les requêtes sur la base de données :

```
public class ClientDAO {
        private static ClientDAO s_Instance;
        private static object s_InstanceLocker = new
object();

        // Singleton
        public static ClientDAO Instance {
                get {
                        lock (s_InstanceLocker) {
                                if (s_Instance == null) {
                                        s_Instance = new
ClientDAO();
                                }
                                return s_Instance;
                        }
                }
        }

        public List<Client> GetClients() {
                List<Client> clients = null;

                using (DbConnection cnx =
Sql.Instance.GetSqlConnection()) {
                        clients = GetClientsFromDB(cnx);
                }

                return clients;
        }

        private List<Client>
GetClientsFromDB(DbConnection cnx) {
                List<Client> clients = null;

                using (DbCommand cmd =
cnx.CreateCommand()) {
```

```
                        cmd.CommandType =
System.Data.CommandType.Text;
                        cmd.CommandText = "SELECT TOP 10
* FROM Person.Contact";
                        using (DbDataReader reader =
cmd.ExecuteReader()) {
                                while (reader.Read()) {
                                        Client client =
new Client();
                                        client.ClientID =
reader["ContactID"] == DBNull.Value ? default(int) :
int.Parse(reader["ContactID"].ToString());

        client.ClientLastName = reader["LastName"] ==
DBNull.Value ? default(string) :
reader["LastName"].ToString();

        client.ClientFirstName = reader["FirstName"] ==
DBNull.Value ? default(string) :
reader["FirstName"].ToString();

        clients.Add(client);
                                }
                        }
                }
                return clients;
        }
}
```

Cette classe, que l'on a appelée ClientDAO (pour Data Access Object) est, elle aussi, composée d'un Singleton. Vous pouvez voir, dans la méthode GetClient, qu'elle récupère l'instance de la classe de connexion à la base de données, puis qu'elle utilise cette connexion pour effectuer une requête. Ici, j'ai utilisé une requête de type texte mais pour bien faire, il aurait fallu utiliser une procédure stockée.

2.4. La couche métier (BLL, Business Logic Layer)

Nous avons donc, d'un coté notre interface graphique et de l'autre, notre couche d'accès au données. Il serait tout à fait possible de relier directement les 2. Cependant, comment feriez-vous si vous aviez besoin d'appliquer des règles ou d'effectuer des opérations sur les résultats issus de la base de données ? Où mettriez-vous votre code ? Pas dans l'interface graphique, car il ne doit y avoir que ce qui concerne l'interface. Dans la couche d'accès aux données ? Non, car cette couche ne traite que de l'accès aux données. Où alors ? C'est dans une nouvelle couche, que l'on appelle BLL (ou Business Logic Layer), que l'on va mettre ce code. C'est le lien entre votre interface utilisateur et votre DAL.

```
public class ClientManager {
        private static ClientManager s_Instance;
        private static object s_InstanceLocker = new object();

        // Singleton
        public static ClientManager Instance {
                get {
                        lock (s_InstanceLocker) {
                                if (s_Instance == null) {
                                        s_Instance = new
ClientManager();
                                }

                                return s_Instance;
                        }
                }
        }

        public ConnectionStringSettings ConnectionString {
                set {
                        Sql.Instance.CnxStringSettings = value;
                }
        }

        public List<Client> GetClients() {
                // Ici, on peut appliquer des règles métier return
ClientDAO.Instance.GetClients();
        }
}
```

Comme vous pouvez le voir, cette classe est, elle aussi, composée d'un Singleton. Et elle possède une méthode (GetClients) qui se contente uniquement d'appeler la méthode de la DAL !

3. Utilisation

A présent, nous avons bien tout ce qu'il nous faut :

- Notre GUI référence notre BLL, nos objets métier, et notre futur couche d'outils
- Notre BLL référence notre DAL, nos objets métier, et notre futur couche d'outils
- Notre DAL ne référence que nos objets métier

Il ne nous reste plus qu'à utiliser tout cela ensemble. Pour cela, il vous suffit, dans le code votre interface graphique, de faire appel aux méthodes de votre BLL :

```
public partial class Window1 : System.Windows.Window {
        public Window1() {
                InitializeComponent();
        }

        private void WindowLoaded(object sender,
RoutedEventArgs e) {
                // Au chargememnt de l'application
                // on indique la chaine de connexion

        BLL.ClientManager.Instance.ConnectionString =
ConfigurationManager.ConnectionStrings["DBConnectionStr
ing"];
        }

        private void btGetClientsClick(object sender,
RoutedEventArgs e) {
                this.lbClients.DataContext =
BLL.ClientManager.Instance.GetClients();
                this.lbClients.DisplayMemberPath =
"ClientLastName";
        }

        private void btUpdateClientNameClick(object
sender, RoutedEventArgs e) {
                //
        }
}
```

4. Gestion des exceptions

Tout ce que l'on a fait à présent est entièrement fonctionnel : vous venez tout juste de développer votre première application utilisant la séparation des couches.

Cependant, une bonne application se doit d'avoir une gestion des exceptions.

C'est à ce niveau là qu'intervient ma fameuse couche « Tools », qui va me permettre, entre autre, de définir mes propres exceptions :

```
public class CustomException : Exception {
	public CustomException()
		: base() {
		//
	}

	public CustomException(string msg)
		: base(msg) {
		//
	}

	public CustomException(string msg, Exception
innerEx)
		: base(msg, innerEx) {
		//
	}
}
```

Il ne nous reste plus qu'à utiliser cette classe dans notre couche métier :

```
public List<Client> GetClients() {
	try {
		// Ici, on peut appliquer des règles
métier
		return ClientDAO.Instance.GetClients();
	} catch (Exception e) {
		throw new CustomException("An error has
occured", e);
	}
}
```

Et lors de l'appel aux méthodes de votre BLL, attrapez les exceptions que vous lancez :

```
private void btGetClientsClick(object sender,
RoutedEventArgs e) {
	try {
		this.lbClients.DataContext =
BLL.ClientManager.Instance.GetClients();
		this.lbClients.DisplayMemberPath =
"ClientLastName";
	} catch (CustomException ce) {
		MessageBox.Show(ce.Message);
	}
}
```

5. Pourquoi développer en utilisant les couches ?

Nous avons vu comment, d'un point de vue technique, nous pouvons mettre en place une application utilisant les différentes couches. Mais une question reste toujours en suspens : « *Pourquoi développer de cette façon ?* ».

En effet, jusqu'à maintenant, vous avez sans doute développé des applications, entièrement fonctionnelles, sans utiliser cette technique. Alors pourquoi ? On pourrait très bien penser que c'est :

- Pour faire joli ? C'est vrai qu'avec une technique comme celle-ci, l'explorateur de solutions de Visual Studio donne tout de suite l'impression d'un travail sérieux, mais je ne suis pas sur que votre employeur préfère le superflu aux résultats
- Pour faire comme les autres ? Oui, mais quel intérêt si vous ne comprenez pas ce que vous faîtes ?

En fait, il y a plusieurs avantages à utiliser cette technique :

- La maintenance des données est indépendante du support physique de stockage
- La maintenance des traitements est simplifiée
- La gestion des traitements depuis la couche de présentation est facilitée
- Le travail en équipe est optimisé
- La migration d'un environnement graphique à un autre est relativement simple

Lorsque l'on dit que le travail en équipe est optimisé, la raison est simple : alors qu'un des membres de l'équipe travaille sur la couche d'accès aux données, un autre peut tout à fait travailler sur la couche métier ou sur l'interface graphique sans perturber le travail de ses collègues.

De même, dans le cas de migration (d'interface utilisateur par exemple), là encore, la tâche est simplifiée. Ainsi, inutile de redévelopper tout ce qui a été fait jusqu'à maintenant : il vous suffit de modifier l'interface.

Dans le cas de notre article, nous sommes partit d'une application WPF pour tout ce qui était interface graphique. Si demain, je souhaitais utiliser la même chose, mais dans une application Web, vais-je devoir tout reprendre ? Bien sur que non : grâce à ce découpage en couche, je serais en mesure de garder la couche d'accès aux données et la couche métier, et de ne changer que la couche GUI (passer de WPF à ASP.NET). Une fois sur ma nouvelle interface graphique, je n'aurais qu'à reprendre les appels à ma BLL (couche métier) et le tour est joué.

6. Conclusion

Comme vous avez pu le voir, développer une application utilisant les différentes couches (DAL, BLL, etc.…) se révèle extrêmement simple à partir du moment où l'on sait exactement ce que l'on doit faire (ce que j'ai tenté de vous démontrer dans cet article). Bien sûr, d'autres concepts pourraient être abordés et certains pourraient être vus plus en détails, peut-être dans un prochain article.

Retrouvez l'article de Thomas Lebrun en ligne : Lien57

Les derniers tutoriels et articles

Premiers pas avec l'API audio OpenAL

Ce tutoriel aborde la programmation audio avec l'API OpenAL. Après avoir vu comment l'installer, nous détaillerons le fonctionnement d'OpenAL et verrons comment jouer très simplement un son. Ce tutoriel est écrit pour la version 1.1 d'OpenAL.

1. Introduction

OpenAL est une API libre (distribuée sous licence LGPL) et multiplateforme offrant une gestion bas niveau de l'audio : sons 3D, flux, capture, sons multi-canaux (jusqu'à 7.1), effets, etc. Ceux qui connaissent OpenGL ne seront pas dépaysés : la philosophie et la syntaxe sont exactement les mêmes.
OpenAL est à la base développé par Loki Entertainment et Creative Labs, mais chaque constructeur de chipset audio peut y ajouter des fonctionnalités propres à son matériel via un système d'extensions.
Utilisée par les plus grands studios de jeux vidéo (Id Software, Epic, ...) OpenAL peut d'ores et déjà être considérée comme un standard sûr et performant.

Pour une présentation plus détaillée, je vous renvoie vers la page d' OpenAL sur Wikipedia (Lien58), ainsi que sur le site officiel (Lien59).

2. Installation et paramétrage

La première chose à faire est de se rendre sur le site officiel (Lien59). Vous y trouverez à peu près tout ce qu'il vous faut : téléchargements, documentation, wiki, ...

Rendez-vous dans la section downloads : vous y trouverez le code source (on peut le laisser de côté pour l'instant), ainsi que les versions de développement d'OpenAL pour les différents systèmes d'exploitation (Windows, Linux, MacOS X). Téléchargez celle qui vous intéresse puis intallez-la. Si vous êtes sous Windows prenez le SDK (13.2 Mo) ; l'installateur lui ne contient que les DLLs OpenAL (une bonne idée est de le fournir avec votre programme, pour les utilisateurs qui n'auraient pas déjà installé OpenAL sur leur machine).
Ne paniquez pas si vous tombez sur une page Creative Labs en cliquant sur un fichier : le fichier en question se trouve au bas de la page, et le téléchargement pourra s'effectuer une fois que vous aurez pris connaissance de la licence.

Le SDK contient tout ce qu'il faut : la documentation de référence, les spécifications de l'API, les fichiers en-tête et bibliothèques, l'installateur, et des exemples en C++ (avec les fichiers projet Visual C++ 7.1 et 8 fournis en prime).

Vous trouverez également sur la page downloads des téléchargements pour ALUT (OpenAL Utility Toolkit), l'équivalent de GLUT pour OpenGL -- pour ceux qui connaissent. Cette bibliothèque contient assez peu de fonctions, principalement pour créer et charger des sons au format WAV depuis un fichier ou depuis des données en mémoire. Ici nous ne l'utiliserons pas, vous êtes donc libre de la télécharger ou non.

Une fois le SDK installé, vérifiez que tout fonctionne correctement en exécutant les exemples fournis dans le répertoire /samples

Une fois les fichiers de développement installés, il faut paramétrer votre environnement de programmation pour qu'il trouve les fichiers d'OpenAL. Pour cela, ajoutez le sous-répertoire /include du SDK aux répertoires de recherche du compilateur, et le sous-répertoire /libs aux répertoires de recherche de l'éditeur de liens. Sous Visual Studio par exemple, cela se trouve dans le menu "Tools", "Options...", "Projects and solutions", "VC++ directories".

3. Initialisation et libération

Nous voilà donc prêts à écrire un premier programme audio avec OpenAL. Première chose à faire : modifier les options de votre projet pour lier avec la bibliothèque OpenAL (OpenAL32.lib sous Windows). Puis il vous faudra inclure les deux en-têtes OpenAL :

```
#include <al.h>
#include <alc.h>
```

<al.h> est l'en-tête principal d'OpenAL, <alc.h> quant à lui définit les fonctions de manipulation du contexte audio. Le contexte est l'environnement qui permet d'exécuter les fonctions audio, et est typiquement très spécifique au système d'exploitation et au driver. Pour ceux qui connaissent OpenGL, c'est là une véritable révolution puisque ALC gère toutes ces spécificités, sans que vous ayiez à écrire une tonne de code non portable comme c'est le cas pour OpenGL. Afin de bien différencier ces deux concepts, les fonctions d'ALC sont préfixées par "alc", et les fonctions d'OpenAL par "al".

Voyons maintenant comment initialiser OpenAL. La première chose est d'ouvrir un device, puis de créer un contexte au sein de ce device. Si vous vous demandez ce que représente le device, on peut le voir comme le "périphérique" qui va être utilisé pour effectuer les sorties audio.

```
bool InitOpenAL(){
    // Ouverture du device
    ALCdevice* Device = alcOpenDevice(NULL);
    if (!Device)
        return false;

    // Création du contexte
    ALCcontext* Context = alcCreateContext(Device,
NULL);
    if (!Context)
        return false;

    // Activation du contexte
    if (!alcMakeContextCurrent(Context))
```

```
        return false;

    return true;
}
```

alcOpenDevice prend en paramètre le nom du device à ouvrir. Ici nous passons NULL pour choisir le device par défaut, mais il est possible de récupérer une liste des devices disponibles via la fonction alcGetString. En effet, si vous lui spécifiez l'option ALC_DEVICE_SPECIFIER, elle vous renverra une liste des devices disponibles sous forme d'une chaîne de caractères terminée par un double caractère nul, chaque device étant séparé par un simple caractère nul.

```
void GetDevices(std::vector<std::string>& Devices){
    // Vidage de la liste
    Devices.clear();

    // Récupération des devices disponibles
    const ALCchar* DeviceList = alcGetString(NULL,
ALC_DEVICE_SPECIFIER);

    if (DeviceList){
        // Extraction des devices contenus dans la
chaîne renvoyée
        while (strlen(DeviceList) > 0){
            Devices.push_back(DeviceList);
            DeviceList += strlen(DeviceList) + 1;
        }
    }
}
```

Revenons à notre code d'initalisation, et plus particulièrement à la création du contexte. alcCreateContext prend en paramètre le device, puis un pointeur sur un tableau d'attributs. Ces derniers sont très peu utiles (en tout cas pour l'instant), nous pouvons donc passer NULL.
Il est également possible de créer plusieurs contextes, mais nous n'en aurons pas l'utilité ici, un seul étant largement suffisant.

Passons maintenant à la libération des ressources. Rien de magique : il s'agit de détruire le contexte, puis le device. Si vous ne les avez pas stockés, il est possible de les récupérer via les fonctions alcGetCurrentContext et alcGetContextsDevice.

```
void ShutdownOpenAL(){
    // Récupération du contexte et du device
    ALCcontext* Context = alcGetCurrentContext();
    ALCdevice*  Device  =
alcGetContextsDevice(Context);

    // Désactivation du contexte
    alcMakeContextCurrent(NULL);

    // Destruction du contexte
    alcDestroyContext(Context);

    // Fermeture du device
    alcCloseDevice(Device);
}
```

4. Charger un son

A présent que nous savons ouvrir et fermer correctement OpenAL, nous allons pouvoir faire des choses intéressantes, à savoir jouer des sons. Mais avant cela il faut les charger : en effet OpenAL ne fournit aucune fonctionnalité pour lire des fichiers audio. Si vous avez téléchargé ALUT vous pourrez grace à lui charger des fichiers au format WAV, mais ici nous allons utiliser une autre bibliothèque : libsndfile. Un peu à l'image de DevIL pour les images, libsndfile permet de charger tout un tas de formats de fichiers audio (wav, raw, aiff, ...). Et pour couronner le tout, elle est gratuite, multiplateforme et open-source.

Rendez-vous sur le site officiel de libsndfile (Lien60), et téléchargez les fichiers nécessaires au développement. Deux options vous sont proposées : télécharger les sources puis les recompiler (tout est fourni pour que ce soit le plus simple possible), ou alors si vous êtes sous Windows, téléchargez directement les fichiers précompilés. Si vous n'avez pas de fichier .lib, référez-vous au fichier readme et suivez les directives pour le générer.

Une fois muni des fichiers nécessaires à l'utilisation de sndfile, nous pouvons écrire une fonction pour charger n'importe quel fichier audio supporté par libsndfile.

Avant d'écrire cette fonction, il faut bien comprendre comment fonctionne OpenAL, et ce que nous allons faire des données chargées. OpenAL repose sur 3 concepts de base : les tampons (sound buffer), les sources (sound source) et l'écouteur (sound listener). Un tampon contient des données audio, ce que l'on appelle habituellement des échantillons. Une source est un moyen de jouer un tampon, avec des propriétés spécifiques (position 3D, volume, amplitude, ...) ; ainsi typiquement on peut placer plusieurs sources dans une scène pour jouer un ou plusieurs tampons audio. Quant à l'écouteur, il est toujours unique et représente l'utilisateur. Vous pouvez lui affecter les mêmes propriétés qu'une source : position 3D, volume, etc.

Pour l'instant nous ne voulons que charger des données audio, ce sont donc les tampons qui vont nous intéresser. A l'instar d'OpenGL, les tampons OpenAL sont identifés par des entiers. Toutes les fonctions relatives aux tampons sont préfixées par alBuffer. Les habitués d'OpenGL reconnaîtront immédiatement les fonctions utilisées ici.

Ici nous commençons par ouvrir le fichier audio avec les fonctions de la bibliothèque libsndfile. Les paramètres sont le nom du fichier, le mode d'ouverture (lecture, écrire, ou lecture / écriture) puis un pointeur vers une structure à remplir avec les informations sur le son.

```
#include <sndfile.h>

ALuint LoadSound(const std::string& Filename){
    // Ouverture du fichier audio avec libsndfile
    SF_INFO FileInfos;
    SNDFILE* File = sf_open(Filename.c_str(), SFM_READ,
&FileInfos);
    if (!File)
        return 0;
```

De ces informations nous pouvons extraire le taux d'échantillonage (nombre d'échantillons à lire par seconde) ainsi que le nombres d'échantillons contenus dans le fichier.

```
    // Lecture du nombre d'échantillons et du taux
d'échantillonnage (nombre d'échantillons à lire par
seconde)
    ALsizei NbSamples  =
static_cast<ALsizei>(FileInfos.channels *
FileInfos.frames);
    ALsizei SampleRate =
static_cast<ALsizei>(FileInfos.samplerate);
```

Nous pouvons donc ensuite lire les échantillons, toujours avec les

fonctions de libsndfile. Ici nous récupérons les échantillons sous forme d'entiers 16 bits signés : il s'agit du format le plus courant -- un peu comme les entiers non signés 32 bits pour les formats de pixels. libsndfile s'occupe de la conversion automatiquement, ainsi même si les échantillons sont stockés sous un autre format dans le fichier audio, vous pourrez les récupérer dans le format voulu.

```
    // Lecture des échantillons audio au format entier 16 bits signé (le plus commun)
    std::vector<ALshort> Samples(NbSamples);
    if (sf_read_short(File, &Samples[0], NbSamples) < NbSamples)
        return 0;
```

Une fois les échantillons récupérés dans un tableau, nous pouvons fermer le fichier, nous n'aurons plus besoin de libsndfile.

```
    // Fermeture du fichier
    sf_close(File);
```

Avant de remplir un tampon OpenAL avec nos échantillons, il faut déterminer le format OpenAL de ceux-ci. De base il n'en existe que 4 :

- AL_FORMAT_MONO8
- AL_FORMAT_STEREO8
- AL_FORMAT_MONO16
- AL_FORMAT_STEREO16

Plus de formats sont disponibles (4.0, 5.1, 6.1, 7.1, ...) mais via le système d'extensions, que nous aborderons plus tard.
Ici ce sera donc AL_FORMAT_MONO16 ou AL_FORMAT_STEREO16, puisque nous avons récupéré nos échantillons sous forme d'entiers 16 bits. Afin de déterminer s'il s'agit d'un son mono ou stéréo, il faut consulter le nombre de canaux, récupérés par libsndfile. 1 canal signifiant un son mono, 2 signifiant un son stéréo.

```
    // Détermination du format en fonction du nombre de canaux
    ALenum Format;
    switch (FileInfos.channels)
    {
        case 1 :  Format = AL_FORMAT_MONO16;   break;
        case 2 :  Format = AL_FORMAT_STEREO16; break;
        default : return 0;
    }
```

Puis vient enfin la création du tampon à proprement parler. Comme déjà précisé, le tampon sera manipulé via un identificateur de type entier non-signé (ALuint).

```
    // Création du tampon OpenAL
    ALuint Buffer;
    alGenBuffers(1, &Buffer);
```

Prenez l'habitude de manipuler les types définis par OpenAL, c'est un gage de portabilité et d'évolutivité.

Une fois le tampon généré à l'aide de alGenBuffers, vous pouvez le remplir avec alBufferData, à qui il faudra donner à manger toutes les données du son que nous avons pris soin de récupérer auparavant : format, échantillons, taille du tableau d'échantillons, et taux d'échantillonnage.

```
    // Remplissage avec les échantillons lus
    alBufferData(Buffer, Format, &Samples[0], NbSamples * sizeof(ALushort), SampleRate);
    // Vérification des erreurs
    if (alGetError() != AL_NO_ERROR)
        return 0;

    return Buffer;
}
```

N'oubliez pas de gérer correctement les erreurs, en appelant alGetError() après un appel de fonction OpenAL. Différents codes d'erreur peuvent être renvoyés, vous pouvez consulter la documentation de référence pour voir lesquels chaque fonction peut générer, ainsi que leur signification.

5. Jouer un son

Afin de jouer un son, il faut créer une source sonore. Une source fera référence à un tampon contenant les échantillons audio à jouer, auquel elle ajoutera des propriétés particulières telles que la position 3D, le volume, l'amplitude, etc.

Les fonctions pour manipuler les sources sont prefixées par alSource. Pour créer une nouvelle source, de la même manière que pour un tampon, il faudra appeler alGenSources.

```
// Création d'une source
ALuint Source;
alGenSources(1, &Source);
```

Nous pouvons ensuite attacher notre tampon à la source fraîchement créée.

```
// On attache le tampon contenant les échantillons audio à la source
alSourcei(Source, AL_BUFFER, Buffer);
```

Notez bien le "i" qui suffixe la fonction alSourcei : il indique que le paramètre passé sera de type entier. Cette convention s'applique à toutes les fonctions OpenAL qui changent / récupèrent un paramètre. Les autres suffixes possibles sont "3i" (3 entiers), "iv" (tableau d'entiers), "f" (flottant), "3f" (3 flottants), et "fv" (tableau de flottants).

Notre source est maintenant prête à jouer le son que nous avons chargé précédemment. Pour se faire, il suffit d'appeler la fonction alSourcePlay.

```
// Lecture du son
alSourcePlay(Source);
```

Il existe d'autres fonctions pour contrôler la lecture : alSourceStop, alSourcePause, et alSourceRewind. Les versions suffixées par "v" de ces fonctions existent également, elles prennent cette fois un tableau de sources pour le cas où vous voudriez effectuer une action sur plusieurs sources simultanément.

Une fois le son lancé, tout ce que nous ferons ici est d'attendre qu'il se termine. Afin de ne pas attendre sans rien faire, nous allons également afficher la position de lecture en secondes.

```
ALint Status;
do{
    // Récupération et affichage de la position courante de lecture en secondes
    ALfloat Seconds = 0.f;
    alGetSourcef(Source, AL_SEC_OFFSET, &Seconds);
    std::cout << "\rLecture en cours... " << std::fixed
```

```
<< std::setprecision(2) << Seconds << " sec";

    // Récupération de l'état du son
    alGetSourcei(Source, AL_SOURCE_STATE, &Status);
}
while (Status == AL_PLAYING);
```

Comme vous le voyez, il est possible de récupérer toute sorte d'informations intéressantes avec alSource (n'oubliez pas le bon suffixe en fonction du type du paramètre !). Ici nous récupérons la position de lecture en secondes avec AL_SEC_OFFSET, mais il aurait également été possible de la récupérer en octets ou en nombre d'échantillons

L'état de lecture de la source est quant à lui récupéré avec AL_SOURCE_STATE. Il existe 4 états : AL_INITIAL, AL_STOPPED, AL_PLAYING, et AL_PAUSED.

Une fois terminé, n'oubliez pas de détruire le tampon et la source.

```
// Destruction du tampon
alDeleteBuffers(1, &Buffer);

// Destruction de la source
alSourcei(Source, AL_BUFFER, 0);
alDeleteSources(1, &Source);
```

Avant de détruire la source n'oubliez pas de détacher le tampon (en mettant sa propriété AL_BUFFER à 0), sans quoi vous pourriez obtenir une erreur.

Enfin, nous avions également parlé tout à l'heure d'un écouteur : ici nous n'avons pas besoin de le définir, les propriétés par défaut étant suffisantes. Il faudra jouer avec celui-ci lorsque vous voudrez gérer des sons 3D (pour représenter position, vitesse et orientation de la caméra), ou pour influer sur le volume global par exemple.

Pour modifier les paramètres de l'écouteur rien de bien compliqué, il faut appeler les fonctions préfixées par alListener :

```
// Définition de la position de l'écouteur (ici
l'origine)
alListener3f(AL_POSITION, 0.f, 0.f, 0.f));

// Définition de la vitesse de l'écouteur (ici nulle)
alListener3f(AL_VELOCITY, 0.f, 0.f, 0.f));

// Définition de l'orientation de l'écouteur (ici il
regarde vers l'axe des Z)
ALfloat Orientation[] = {0.f, 0.f, 1.f, 0.f, 1.f, 0.f};
alListenerfv(AL_ORIENTATION, Orientation));
```

6. Les extensions

Tout comme OpenGL, OpenAL s'est fendu d'un système d'extensions afin de pouvoir supporter les fonctionnalités particulières de chaque constructeur de chipset audio, et de pouvoir évoluer sans avoir à sortir une nouvelle version du SDK à chaque modification. En gros, plutôt que d'utiliser une fonction / constante définie dans al.h, vous irez la demander directement au driver de manière dynamique avec les fonctions qui vont bien. Attention, il faudra bien distinguer les extensions propres au contexte (préfixées par ALC) du reste (préfixées par AL).
La liste des extensions disponibles classées par systèmes peut être consultée sur le site officiel : Lien61.

Afin de lister toutes les extensions supportées par votre système, vous pouvez utiliser alGetString avec l'option AL_EXTENSIONS.

```
const ALchar* Extensions = alGetString(AL_EXTENSIONS);
```

Pour vérifier si une extension est bien supportée pas la peine de parcourir cette chaîne : la fonction alIsExtensionPresent le fera pour vous.

```
bool IsMultiChannelSupported =
(alIsExtensionPresent("AL_EXT_MCFORMATS") == AL_TRUE);
```

Lorsqu'une extension est supportée, elle est généralement accompagnée de fonctions et / ou constantes, qu'il faudra aller chercher dynamiquement via les fonctions alGetProcAddress et alGetEnumValue. alGetProcAddress permet d'obtenir un pointeur vers une fonction, et alGetEnumValue permet de récupérer la valeur d'une constante.

Par exemple, si vous voulez gérer les formats ayant plus de deux canaux, vous pouvez récupérer des formats supplémentaires (si votre environnement le supporte) :

```
ALenum Format = 0;
switch (FileInfos.channels){
    case 1 : Format = AL_FORMAT_MONO16;
break;
    case 2 : Format = AL_FORMAT_STEREO16;
break;
    case 4 : Format =
alGetEnumValue("AL_FORMAT_QUAD16");  break;
    case 6 : Format =
alGetEnumValue("AL_FORMAT_51CHN16"); break;
    case 7 : Format =
alGetEnumValue("AL_FORMAT_61CHN16"); break;
    case 8 : Format =
alGetEnumValue("AL_FORMAT_71CHN16"); break;
}
```

Les extensions ALC (relatives au contexte donc) sont manipulées exactement de la même manière, excepté qu'il faudra utiliser les fonctions préfixées par "alc" plutôt que "al" (alcIsExtensionPresent, alcGetProcAddress, alcGetEnumValue).

7. Conclusion

OpenAL est est une vraie révolution au niveau de la programmation audio : elle permet de développer de manière abordable des applications audio de qualité sans se soucier des soucis de portabilité, et en évitant d'avoir recours à des moteurs pas toujours gratuits comme FModEx ou Bass. De plus elle sera très facile d'accès pour les développeurs qui utilisent OpenGL, comme vous avez pu le constater.

Pour ceux qui voudraient aller plus loin, d'autres tutoriels suivront abordant la lecture de flux ou encore la capture audio.

Le code source complet de ce tutoriel est disponible, avec les fichiers projets pour Visual Studio 2005 : openal-src.zip (201 Ko) (Lien62)

Si vous recherchez des sons ou musiques gratuits pour vos développements, pensez à faire un tour par notre page de ressources gratuites ! (Lien63)

Si vous avez des suggestions, remarques, critiques, si vous avez remarqué une erreur, ou bien si vous souhaitez des informations complémentaires, n'hésitez pas à me contacter !

La FAQ SDL

Pourquoi certaines fonctions provoquent des erreurs ?

Si un appel SDL est fait avant l'appel à **SDL_Init**, le comportement est indéfini. Il faut faire attention aux appels SDL qui se trouvent dans les constructeurs puisque, souvent, l'initialisation de la SDL se fait dans la fonction main et si un objet est déclaré en global, le constructeur sera appelé avant la première instruction du main.

Pour éviter ce genre de problème, ne mettez pas d'appels SDL dans les constructeurs.

Où se trouvent les sorties standard et d'erreur sous Windows ?

Sous Windows, la bibliothèque SDL redirige respectivement la sortie standard et la sortie d'erreur vers les fichiers **stdout.txt** et **stderr.txt** respectivement. A la fin de l'exécution, la bibliothèque vérifie si les fichiers sont vides et, si c'est le cas, la SDL les efface.

Comment se servir de SDL_Flip ?

Lorsqu'un code SDL est écrit, il posséde généralement une boucle globale qui inclut une boucle événementielle, ceci permet de bien définir l'emplacement de l'appel à SDL_Flip :

Boucle générale d'un programme SDL

```
while( jeuencours ) {
	while( SDL_PollEvent(&event) ) {
		case SDL_QUIT :
			jeuencours = 0;
			break;
		default:
	}

	/* Code de rendu */

	..... Pas de SDL_Flip ....

	/* Un seul SDL_Flip */
	SDL_Flip(ecran);
}
```

Remarque : Il ne faut pas voir SDL_Flip comme une solution pour afficher le dernier Blit qui vient d'être fait. Il faut un et un seul appel par code de rendu.

A quoi sert SDL_Flip ?

Lorsqu'on utilise un double tampon pour l'affichage, le programme dessine la prochaine image en même temps que la carte graphique affiche l'image précédente.

Supposons que la carte graphique affiche le tampon A et que le code de rendu travaille sur le tampon B.

SDL_Flip permet de dire à la carte graphique que le code de rendu sur le tampon B est fini et qu'elle peut à présent l'afficher.

Ainsi, le code de rendu qui va suivre va écrire sur le tampon A pendant que la carte graphique affiche l'image qui se trouve dans le tampon B.

Le prochain SDL_Flip fera que la carte graphique affichera tampon A pendant que le code de rendu travaille sur le tampon B.

Comment modifier la position par défaut des fenêtres ?

Pour modifier la position de la fenêtre par défaut, on peut ajouter ce code :

Le changement de position de la fenêtre

```
#include <SDL/SDL_getenv.h>

putenv("SDL_VIDEO_WINDOW_POS=center"); //pour centrer la fenêtre
putenv("SDL_VIDEO_WINDOW_POS=0,0");    //pour placer en x,y (ici 0,0)
```

Attention, ceci ne fonctionnera pas avec tous les systèmes d'exploitation mais il fonctionne normalement au moins sous Windows et Linux.

Retrouvez ces questions et de nombreuses autres sur la FAQ SDL : Lien65

Les derniers tutoriels et articles

Protéger son code source PHP avec bcompiler

Il est possible de protéger son code sources PHP des petits malins qui pourraient le récupérer car parfois on peut vendre ou donner des scripts mais on ne souhaite pas que le destinataire possède les sources. La bibliothèque Bcompiler vous aidera dans cette tâche.

1. Bcompiler c'est quoi ?

1.1. Description

Bcompiler est une bibliothèque php crée à l'origine :

- Pour encoder un script complet dans une application PHP propriétaire
- Pour encoder des classes et/ou des fonctions dans une application PHP propriétaire
- Pour permettre d'utiliser des applications PHP-GTK sur des bureaux clients sans avoir besoin du fichier php.exe.
- Pour rendre faisable de convertir un code PHP en C

Nous resterons que sur le premier but. Il faut activer l'extension php_bcompiler.dll sous Windows ou télécharger, décompresser et recompiler PHP sous linux (explication ici : Lien66)

1.2. Comment ça marche ?

En réalité Bcompiler transforme votre code en ByteCode (similaire à Java ou C#) non lisible par un utilisateur mais seulement par PHP, il s'agit d'un code intermédiaire plus abstrait que le code machine non directement exécutable. Il est contenu dans un fichier binaire un peu plus lourd qui représente un script, tout comme un fichier objet produit par un compilateur, ce ByteCode est directement interprété par PHP et il n'est pas possible de retrouver le code PHP original.

2. Avec un exemple

2.1. Pré-requis

Imaginons deux scripts hello.php et function.php que nous voulons protéger.

hello.php

```
<?php
include "function.php";
echo hello();
?>
```

function.php

```
<?php
function hello()
{
        return "hello";
}
?>
```

2.2. Transformation du code

Tout d'abord il faut transformer nos deux scripts en ByteCode, nous allons passer par un script en PHP qui permettra de les transformer.

transform.php

```
<?php

$fh = fopen("hello.phb", "w");

bcompiler_write_header($fh);

bcompiler_write_file($fh, "hello.php");

bcompiler_write_footer($fh);

fclose($fh);

?>
```

2.2.1. Explications

On crée par cette transformaiton le fichier hello.phb

```
$fh = fopen("hello.phb", "w");
```

On écrit dans ce fichier un en-tête de type Bcompiler pour que l'interpréteur PHP puisse comprendre qu'il s'agit de ByteCode.

```
bcompiler_write_header($fh);
```

On écrit dans ce fichier le Byte Code du script hello.php

```
bcompiler_write_file($fh, "hello.php");
```

On écrit le pied du script

```
bcompiler_write_footer($fh);
```

Et on ferme notre fichier

```
fclose($fh);
```

Et on fait la même chose avec function.php pour créer function.phb.

II-B-2. Résultat
Maintenant hello.phb contient notre script hello.phb en ByteCode :

Comme vous pouvez (ou presque) le voir, le code source est devenu illisible.

3. Finition

Maintenant que l'on a nos deux scripts hello.phb et function.phb il faut modifier les fichiers hello.php et function.php pour qu'ils puissent appeler leurs homologues 'ByteCodé'.

hello.php
```
<?php
include "hello.phb";
?>
```

function.php
```
<?php
include "function.phb";
?>
```

L'appel par include "mon_fichier_ByteCodé.phb" permet d'exécuter le ByteCode, donc sur la page hello.php on verra hello.

4. Conclusion

Il faut éviter de faire de multiples includes de fichers car cela ralentit considérablement l'exécution des scripts. Cependant, il existe d'autre fonction de cette extension qui permettent ce genre d'opération (http://us2.php.net/manual/fr/ref.bcompiler.php).

Bcompiler est une bonne façon de protéger son code et offre de nouvelle perspective pour PHP, de plus il est gratuit et offre une excellente alternative à tous les autres logiciels (payants) proposant les mêmes services.

Retrouvez l'article de Maxime Ohayon en ligne : Lien64

Les livres sécurité

Sécurité des réseaux

A l'heure des communications électroniques universelles, menaces et fraudes entravent plus que jamais la productivité et la sécurité des entreprises et des individus. Heureusement la sécurité des réseaux a mûri, conduisant au développement d'applications et de techniques de protection efficaces.
Cet ouvrage fournit une vision d'ensemble des pratiques et des principes de sécurité vitaux pour le traitement de tout échange de données sur un réseau.
Il développe les points clé nécessaires à la mise en œuvre d'une politique de sécurité moderne :
- la cryptographie, qui constitue le fondement de nombre d'applications de sécurité : chiffrement, fonctions de hachage, signatures numériques, échange de clés ;
- les outils et applications destinés à la sécurité des réseaux, dont Kerberos, les certificats X5090, PGP, S/MIME, la sécurité If, SSI./TI.S, SET, etc. ;
- la gestion de réseaux (SNMPO) et la sécurité web ;
- la sécurité au niveau système, incluant un exposé des menaces causées par les attaques d'intrus, de vers et virus, et les mesures à prendre pour les combattre, ainsi que l'utilisation des pare-feu et des systèmes sécurisés;
Une abondante bibliographie et des liens vers des sites web consacrés à la sécurité des réseaux complètent cet ouvrage.

Critique du livre par la rédaction (bob)

Ce livre se veut très général et aborde les grands problèmes de sécurité actuels. Il présente une approche très universitaire de la sécurité des réseaux, et pourrait donc convenir comme une introduction à un cours de sécurité plus approfondi. Le problème est qu'il ne fait qu'effleurer les différents thèmes abordés et laisse un grand nombre de questions en suspend.
Des points importants comme la cryptographie, IPSec ou encore SSL sont présentés. Ceci permet de donner un bon aperçu des techniques employées actuellement pour sécuriser les réseaux. Encore une fois, les notions présentées ne sont pas suffisantes pour permettre à un administrateur de prendre des décisions concrètes ou de mettre en place un système de sécurité efficace.
L'ouvrage sensibilise donc aux problèmes de sécurité et montre un certain nombre de techniques qui peuvent être utilisées, mais on regrette qu'il reste aussi vague.
L'introduction sur la cryptographie est complète et intéressante, ce qui est un très bonne chose étant donné l'importance de la cryptographie dans la sécurité actuelle, que ce soit des données ou des réseaux. La partie cryptographie est à mon avis suffisante pour permettre à une personne nouvelle à ces techniques de comprendre le fonctionnement général des applications cryptographiques. Cependant il est probablement nécessaire de livre un ouvrage consacré entièrement à la cryptographie si l'on s'intéresse sérieusement à la sécurité des réseaux étant donné l'importance de cette notion.
Les standards présentés sont assez peu nombreux. Kerberos, IPSec, SSL, PGP, sont certes des outils importants pour la mise en place d'un réseau sécurisé, mais on ne peut pas résumer la sécurité des réseaux à ces seuls outils. Kerberos, dont le fonctionnement est très détaillé (et pour différentes versions), aurait peut être pu être présenté plus rapidement au profit d'explications supplémentaires sur la mise en place concrète d'un réseau utilisant Kerberos. Un explication brève suivie d'un nombre conséquent d'exemples sur le fonctionnement aurait à mon avis été préférable. En effet, les explications parfois complexes sur les technologies utilisées rendront cet ouvrage difficile à lire pour un débutant, publique pourtant visé par cet ouvrage.

Commentaire personnel :
J'ai trouvé trouvé l'ouvrage relativement difficile à lire et souvent décevant du fait qu'il ne fait que décrire quelques standards. La sécurité d'un réseau est très complexe et la connaissance du principe de fonctionnement de quelques standards ne suffira pas à mettre en place un réseau protégé efficacement. Pour cela il est nécessaire que le lecteur puisse voir des exemples plus concrets de réseaux complets utilisant ces technologies, et comment elles-ont été mises en place.

Retrouvez ce livre dans la rubrique sécurité : Lien68

Cryptographie en pratique

La sécurité informatique devient une condition essentielle de la prospérité des entreprises, de leur expansion et à terme de leur survie. La cryptographie est une promesse de sécurité dans un monde en réseau mais, assez curieusement, des ouvrages manquent qui décriraient concrètement comment implémenter la cryptographie et l'intégrer à des systèmes réels. Ce livre se veut donc le premier manuel de cryptographie clé en mains, faisant le lien entre cryptographie théorique et applications cryptographiques. Il nous plonge directement dans le "comment faire" en fournissant explications, règles et recettes sur notamment : Comment choisir les primitives cryptographiques, des chiffrements par bloc aux signatures électroniques ? Comment implémenter les systèmes et algorithmes cryptographiques de façon à garantir la sécurité sur les ordinateurs actuels ? Comment construire une architecture cohérente qui garantisse pour chaque élément du système le niveau de sécurité requis ? Comment placer la sécurité au cœur du projet et pourquoi elle affecte toutes les partie du système ? Comment accroître la sécurité d'un système tout en diminuant sa complexité en utilisant des interfaces cryptographiques simples ?

Critique du livre par la rédaction (Cecile Muno)

Un livre très intéressant et touchant à l'ensemble des phénomènes liés à la sécurité des données. Pour chaque élément décrit, le schéma est toujours le même ce qui permet une comparaison aisée des caractéristiques des méthodes étudiées.

Le livre commence par une analyse exhaustive des problèmes de sécurité en poussant de plus en plus lin les possibilités de cassage de ces sécurités.

Seule ombre dans le shcéma de cet ouvrage, le peu d'exemples concrets qui permettraient d'encore mieux appréhender les explications très techniques. Le rythme évolutif des codages permet à tous de plonger à son propre niveau dans ce domaine très spécifique.

Retrouvez ce livre dans la rubrique sécurité : Lien69

Configuring IPCop Firewalls

Ce livre décrit les étapes nécessaires pour mettre en place la solution IPCop de parefeu entre votre point d'accès internet et votre réseau. Après une partie très générale sur les logiciels libres et une explication rapide de la conception entièrement libre de IPCop, le livre va vous entraîner depuis l'installation jusqu'à la mise en service de votre projet. Les chapitres vous guideront pas à pas de la préparation à votre première installation. On vous explique ensuite la configuration des divers éléments contenus dans IPCop, de la configuration basique à une approche plus fine des services présents, tels que le VPN. Le livre vous entraîne ensuite dans différents tests pour évaluer votre mise en place, pour terminer sur le dernier élément de votre installation : la maintenance.

Critique du livre par la rédaction (Katyucha)

Lors de ma lecture, je me suis étonné d'être aussi bien guidé dans l'installation d'IPCop avec toujours en toile de fond une explication réaliste sur le domaine de la sécurité. L'auteur ne s'enflamme pas sur les possibilités du produit : IPCop est un firewall pour la maison et les bureaux de petites tailles.
J'avoue que les premieres pages généralistes sur la licence GPL et la présentation des produits libres m'ont ennuyé. Si vous êtes un habitué du monde libre, je ne pense pas que ce passage vous soit fort utile. Par contre, pour les néophytes, je trouve qu'il est clairement et simplement rédigé.
L'ensemble des explications sont cohérentes et données au moment opportun.
La partie configuration est très bien expliquée. Dans beaucoup de livres techniques, on arrive souvent à se perdre dans des explications, des études de cas mais, ici, l'auteur a su rester clair et concis dans ses descriptions. Une personne débutante peut vraiment se sentir à l'aise dans toutes les opérations.

Le livre, bien qu'écrit en anglais, se lit très bien pour les habitués de l'anglais technique. L'auteur écrit dans un style très simple sans fioriture et en fait, on pourrait mettre tout le livre dans un man (d'accord, le man serait assez gros).

La dernière pensée que j'ai eue au sujet de ce livre, c'est pourquoi acheter ce livre alors qu'il y a une grande quantité de documentation sur cette distribution. Je dirais la sérénité d'une explication limpide et une documentation unique et complète. Ce livre est un document de référence à conserver dans la bibliothèque de votre bureau, si vous désirez installer cette solution.

Retrouvez ce livre dans la rubrique sécurité : Lien70

Liens

Lien1 : http://gfx.developpez.com/tutoriel/java/nio/
Lien2 : http://gfx.developpez.com/tutoriel/java/ioc/
Lien3 : http://java.developpez.com/livres/?page=Francais#L2841774112
Lien4 : http://java.developpez.com/livres/?page=Francais#L2841774112
Lien5 : http://www.freebsd.org/
Lien6 : http://www.pa.msu.edu/~tigner/bsddvd.html
Lien7 : http://www.freebsd.org/cgi/man.cgi?query=make
Lien8 : http://www.freebsd.org/cgi/man.cgi?query=mdconfig
Lien9 : http://www.freebsd.org/cgi/man.cgi?query=mount
Lien10 : http://www.freebsd.org/cgi/man.cgi?query=tar
Lien11 : http://www.freebsd.org/cgi/man.cgi?query=rm
Lien12 : http://www.freebsd.org/cgi/man.cgi?query=mkisofs&apropos=0&sektion=0&manpath=FreeBSD+6.1-RELEASE+and+Ports&format=html
Lien13 : http://www.freebsd.org/cgi/man.cgi?query=growisofs&manpath=FreeBSD+6.1-RELEASE+and+Ports
Lien14 : http://oregnier.developpez.com/cours/unix/freebsd/makedvd/
Lien15 : http://linux.developpez.com/livres/?page=livresDIST#L2746030101
Lien16 : http://linux.developpez.com/livres/?page=livresNET#L2212114451
Lien17 : http://linux.developpez.com/livres/?page=livresSYS#L2841771415
Lien18 : http://devzone.zend.com/node/view/id/119
Lien19 : http://g-rossolini.developpez.com/tutoriels/php/zend-framework/debuter
Lien20 : http://morpheus.developpez.com/dlldotnet
Lien21 : http://www.microsoft.com/france/msdn/vcsharp/Utilisez-Pinvoke.mspx
Lien22 : http://msdn2.microsoft.com/en-us/library/aa969540.aspx
Lien23 : http://msdn2.microsoft.com/fr-fr/library/system.runtime.interopservices.dllimportattribute.preservesig(VS.80).aspx
Lien24 : http://msdn2.microsoft.com/fr-fr/library/system.windows.forms.control.wndproc(vs.80).aspx
Lien25 : http://badger.developpez.com/tutoriels/dotnet/effet-glass-vista/
Lien26 : http://c.developpez.com/faq/vc/?page=Conversions#ConvertCString
Lien27 : http://msdn2.microsoft.com/en-us/library/c7t43w0s%28VS.80%29.aspx
Lien28 : http://c.developpez.com/faq/vc/?page=DoDataExchange#UpdateData
Lien29 : http://sourceforge.net/project/showfiles.php?group_id=7586
Lien30 : http://www.boost-consulting.com/
Lien31 : http://arb.developpez.com/c++/boost/install/vc++/
Lien32 : http://msdn2.microsoft.com/en-us/library/system.convert.aspx
Lien33 : http://msdn2.microsoft.com/en-us/library/system.formatexception.aspx
Lien34 : http://c.developpez.com/faq/vc/?page=Conversions#CStringFormat
Lien35 : http://msdn2.microsoft.com/en-us/library/dwhawy9k.aspx
Lien36 : http://farscape.developpez.com/Articles/Conversions/
Lien37 : http://jab.developpez.com/tutoriels/db2/sqlrecursif
Lien38 : http://fr.wikipedia.org/wiki/BackTrack
Lien39 : http://www.red-database-security.com/security_training/oracle_security_training.html
Lien40 : http://www.petefinnigan.com
Lien41 : http://www.oracle.com/technology/deploy/security/index.html
Lien42 : http://oracle.developpez.com/faq/?page=4-3
Lien43 : http://fadace.developpez.com/oracle/schema
Lien44 : http://oracle.developpez.com/faq/?page=4-3#toolnoconnect
Lien45 : http://fadace.developpez.com/oracle/securite/
Lien46 : http://miles.developpez.com/tutoriels/hardware/config/
Lien47 : http://baptiste-wicht.developpez.com/tutoriel/windows/demarrage/
Lien48 : http://windows.developpez.com/livres/#L274402130X
Lien49 : http://windows.developpez.com/livres/#L2746034867
Lien50 : http://gfx.developpez.com/tutoriel/mac/quartzcomposer
Lien51 : http://macrabbit.com/cssedit/
Lien52 : http://www.acqualia.com/picturesque/
Lien53 : http://blog.developpez.com/index.php?blog=142
Lien54 : http://ymarec.developpez.com/tutoriel/rails/initiation/
Lien55 : http://pbnaigeon.developpez.com/tutoriel/rails/ajax-facile-RJS
Lien56 : http://ruby.developpez.com/livres/#L2841773884
Lien57 : http://morpheus.developpez.com/architecture/
Lien58 : http://fr.wikipedia.org/wiki/OpenAL
Lien59 : http://www.openal.org/
Lien60 : http://www.mega-nerd.com/libsndfile/
Lien61 : http://www.openal.org/extensions.html
Lien62 : http://loulou.developpez.com/tutoriels/openal/premiers-pas/fichiers/openal-src.zip
Lien63 : http://jeux.developpez.com/medias/
Lien64 : http://loulou.developpez.com/tutoriels/openal/premiers-pas/
Lien65 : http://jeux.developpez.com/faq/sdl
Lien66 : http://us2.php.net/manual/fr/ref.bcompiler.php
Lien67 : http://maxime-ohayon.developpez.com/tutoriels/bcompiler/
Lien68 : http://securite.developpez.com/livres/securite/#L2711786536
Lien69 : http://securite.developpez.com/livres/securite/#L2711748200
Lien70 : http://securite.developpez.com/livres/securite/#L1904811361
Lien71 : http://zend-framework.developpez.com/