Developpez Magazine

Edition de Aout-Septembre 2007.
Numéro 11.
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

Quoi de mieux qu'une séléction d'articles de Developpez.com pour la rentrée ? Developpez.com vous offre le 11ème numéro de son magazine pour cet été.

Retrouvez comme d'habitude de nombreux tutoriels, des critiques de livres, des questions/réponses, ...

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

La rédaction

Article PHP

Commencer avec symfony

symfony est un framework libre pour PHP5. S'inspirant de Ruby On Rails, ce framework s'appuie sur un modèle MVC (Modèle-Vue-Contrôleur). Il implémente également l'Ajax. Découvrez comment développer avec simfony.

par **Christopher Maneu**
Page 9

Article Windows

Activer le SSL dans IIS 7.0 sous Vista

Cet article va vous permettre d'activer le protocole SSL dans IIS 7.0 sans pour autant devoir se procurer un certificat en bonne et due forme auprès d'une autorité de certification reconnue ce qui est onéreux et pas forcément indispensable dans un environnement de développement.

par **Ronald Vasseur**
Page 42

La covariance dans le JDK1.5

La version 5 du JDK (Tiger) a introduit une notion qui est passée inaperçue face aux autres nouveautés (généricité, auto-boxing, etc...) Cette nouveauté est la covariance des types retours. Cet article aura pour but de présenter les concepts qu'introduit cette notion. Je vous proposerai quelques exemples permettant de mettre en évidence son utilité.

1. Rappel

La redéfinition ou la surcharge suivent des règles précises dans les relations d'héritage.
Dans le cadre de la redéfinition d'une méthode, il faut que la signature de la méthode de la classe fille ait exactement la même signature.

Redéfinition
```
//Classe Mère
public String maMethod(String a,String b){//...}
//Classe fille
public String maMethod(String a,String b){//...}
```

Dans le cadre de la surcharge d'une méthode, la méthode de la classe fille se distinguera par ses paramètres (leurs nombres et leurs types).

Surcharge
```
//Classe Mère
public String maMethod(String a,String b){//...}
//Classe fille
public String maMethod(String a,Integer c){//...}
```

2. La covariance

Dans les versions antérieures au JDK 1.5 ces deux règles étaient valables. Depuis la version 1.5 et supérieures ces règles ont légèrement changées.

En effet, il est maintenant possible de modifier le type retour d'une méthode que l'on redéfinit. Dans un premier temps, cette nouvelle possibilité peut paraître quelque peu déstabilisante.

Après réflexion, la covariance des types retour donne la possibilité de s'affranchir de la conversion de type explicite (cast). Le cast, une technique qui introduit des faiblesses dans les programmes et qui met en évidence une faiblesse du typage du langage Java.

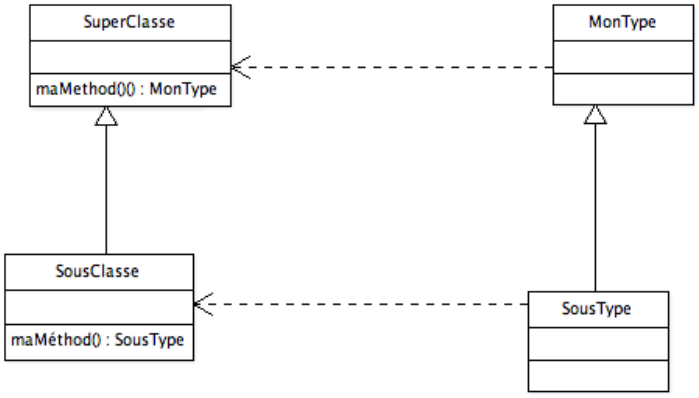
Une contrainte existe dans la mise en oeuvre de cette technique, il faut que le nouveau type retour soit un sous-type du type déclaré dans la super classe. Cette contrainte reste logique et cohérente. En effet, l'héritage raffine (spécialise) le comportement d'une classe, il est logique que les méthodes redéfinies dans les sous-classes spécialisent aussi les types retours de ses méthodes.

L'utilisation de la covariance ne doit pas être systématique, il faut que celle-ci présente un réel intérêt.

Dans la suite de l'article, nous allons voir plusieurs cas d'utilisation de ce [nouveau] principe qui est maintenant autorisé.

3. Le principe

Nous allons voir ici un exemple simple.



Dans cet exemple, le type de retour a été spécialisé (MonType -> SousType)
Entre les deux, il y a une relation de sous type. SousType est un sous type de MonType.

Cet exemple n'est peut-être pas très parlant et on ne saisit pas forcement l'intérêt de faire cette modification.
Ne vous inquiétez pas, dans la suite de l'article nous verrons un exemple concret permettant de voir l'intérêt de cette pratique.

4. La mise oeuvre

Ci-dessous nous allons voir un exemple plus parlant avec la méthode clone() définie par l'interface Cloneable. Lorsque l'on souhaite cloner en profondeur un objet on implante cette interface et on redéfinit la méthode clone().

Voila la signature de la méthode :
```
public Object clone() throws
CloneNotSupportedException;
```

On notera que cette méthode renvoie un type Object. Si l'objet à cloner est de ce type aucun problème. Par contre, si le l'objet renvoyé n'est pas de ce type, il faudra caster l'objet renvoyer dans le type de l'objet cloné.

```
package covariance;

public class ClassCloneable implements Cloneable {
	private String objecName = null;

	public String getObjectName() {
		return objecName;
	}
```

```
	public void setObjectName(String objecName) {
		this.objecName = objecName;
	}

	 @override
	public Object clone() throws
CloneNotSupportedException {
	ClassCloneable newRef = (ClassCloneable)
super.clone();

		return newRef;
	}
}

public class Main {
    public static void main(String[] args) {
	ClassCloneable classCloneable = new
ClassCloneable();
	 classCloneable.setObjectName("nouvel Objet
Name");
	 try{
		AutreClassCloneable newRef =
(AutreClassCloneable)classCloneable.clone();
	 }catch(Exception err){
		err.printStackTrace();
	 }
}
```

Dans cet exemple, nous voyons que nous devons passer par un 'cast' explicite de l'objet renvoyé par la méthode clone.
Nous sommes obligés de passer par là afin de retomber sur le type d'instance de l'objet cloné.

Cette pratique peut comporter des risques. Dans le cas ci-dessus le code se compilera et s'exécutera correctement.
En revanche, un développeur peu scrupuleux (ou tête en l'air) aurait pu écrire le code suivant :
package covariance;

```
public class ClassCloneable implements Cloneable{
	private String objecName = null;

	public String getObjectName() {
		return objecName;
	}

	public void setObjectName(String objecName) {
		this.objecName = objecName;
	}

	 @override
	public Object clone() throws
CloneNotSupportedException {
		ClassCloneable newRef = (ClassCloneable)
super.clone();
		return newRef;
	}
}

package covariance;

public class AutreClassCloneable implements Cloneable{
	private String objectName = null;

	public String getAutreObjectName() {
		return objectName;
	}

	public void setAutreObjectName(String objectName)
{
		this.objectName = objectName;
	}

	 @override
	public Object clone() throws
CloneNotSupportedException {
		ClassCloneable newRef =
(ClassCloneable) super.clone();
		return newRef;
	}
}

public class Main {
    public static void main(String[] args) {
	ClassCloneable classCloneable = new
ClassCloneable();
	 classCloneable.setObjectName("nouvel Objet
Name");
	 try{
		AutreClassCloneable newRef =
(AutreClassCloneable)classCloneable.clone();

	 }catch(Exception err){
		err.printStackTrace();
	 }
}
```

Dans l'exemple ci-dessus, la compilation se passera sans problème. Le type retour de la méthode est bien un sous-type de la classe Object.

En revanche, au moment de l'exécution, la liaison tardive posera quelques problèmes de 'cast'. Le type renvoyé par la méthode est ClassCloneable et on veut le transtyper en AutreClassCloneable. Il n'y a aucune relation de type entre ces deux classes. L'exécution de ce programme aboutira à une exception de type ClassCastException.

Ce type de situation montre une faiblesse du typage en Java. En effet, le fait de devoir forcer le typage d'un objet peut donner lieu à des problèmes qui interviennent uniquement à l'exécution.

L'utilisation de la covariance sur les types retours nous permet de sécuriser notre code.

Nous allons reprendre l'exemple vu précédemment en introduisant la covariance sur le type retour de la méthode clone.
package covariance;

```
public class ClassCloneable implements Cloneable{
	private String objecName = null;

	public String getObjectName() {
		return objecName;
	}

	public void setObjectName(String objecName) {
		this.objecName = objecName;
	}

	@override
	protected ClassCloneable clone() throws
CloneNotSupportedException {
			return (ClassCloneable)
super.clone();
	}
}
```

Dans cette nouvelle implémentation de la classe ClassCloneable, nous pouvons remarquer que le type de retour de la méthode clone() a été spécialisé.

Malgré la modification de la signature cette méthode redéfinit bien la méthode clone de l'interface Cloneable.

5. La covariance : un exemple concret

Nous allons voir dans ce paragraphe un cas d'utilisation concret mettant en évidence l'intérêt de la covariance.

Ci-dessous une interface représentant un panier ainsi que deux implémentations de celle-ci :

```
public interface Bag {
    public void addItem(Item i);

    public Bag addItems(Bag b);
}

//Premiere implantation l'interface Bag

public class BooksBag implements Bag {
    public void addItem(Item i) {
        //code d'ajout d'un item au panier'
    }

    public void printTitles(){
        //code d'affichage du titre de chaque livre
    }

    public Bag addItems(Bag b) {
        //Code d'ajout des items au panier
        return this;
    }
}

//Deuxième implantation du panier.

public class FlowersBag implements Bag{
    public void addItem(Item i) {

    }

    public void compterNombreDePetale(){
        //Code permettant de compter le nombre de pétale
    }

    public Bag addItems(Bag b) {
        //Code d'ajout au panier
       return this;
    }
}
```

Dans cette première version, nous n'utilisons pas le principe de covariance.
En effet, la méthode susceptible d'appliquer ce principe est addItems :Bag->Bag. Nous allons mettre en évidence les problèmes sous-jacents dans ce cas.

Ici nous mettons en évidence les problèmes que l'on peut rencontrer :

```
public class Main {
        public static void main(String[] args) {
                BooksBag bbAutres = new BooksBag();
                FlowersBag fbAutres = new FlowersBag();

                BooksBag bb = new BooksBag();(1)
                FlowersBag fb = new FlowersBag();(2)

                BooksBag bag =
(BooksBag)bb.addItems(bbAutres); (3)
                FlowersBag fbag =
(FlowersBag)fb.addItems(fbAutres); (4)

                FlowersBag bag =
(FlowersBag)bb.addItems(bbAutres); (5)
                BooksBag fbag =
(BooksBag)fb.addItems(fbAutres); (6)
        }
}
```

Nous commençons par instancier un objet de type BooksBag et un objet de FlowersBag (lignes 1 et 2).
Ensuite, nous appliquons la méthode addItems sur les deux objets instanciés précédemment (ligne 3 et 4).

La remarque qui peut être faite concerne la nécessiter de caster les deux objets dans leur type respectif
lors de la récupération de la référence de l'objet modifié. Ce qui n'est pas une bonne chose.

Remarques : Il est vrai que nous pourrions être plus abstrait et utiliser le type de l'interface Bag cela nous éviterait d'utiliser le mécanisme de cast.
En faisant ça, nous perdrions de l'information pour chaque type :

- La méthode *compterNombreDePetale* pour la classe **FlowersBag**
- La méthode *printTitles* pour la classe **BooksBag**

On peut constater aussi qu'il est possible de faire des erreurs de typages (ligne 5 et 6). Ici, la compilation se passera sans problème; par contre, une exception de type ClassCastException sera levée lors de l'exécution.

Maintenant que nous avons mis en évidence ce problème, voyons comment la covariance va nous aider à obtenir un code plus propre,
c'est-à-dire que l'on pourra se passer du mécanisme de cast et surtout d'interdire dès la compilation le mélange de type comme vu ci-dessus.

Nous allons maintenant utiliser le mécanisme de covariance afin d'éviter les problèmes vu précédemment :

```
public interface Bag {
    public void addItem(Item i);

    public Bag addItems(Bag b);
}

//Premiere implantation l'interface Bag
public class BooksBag implements Bag {
    public void addItem(Item i) {
        //code d'ajout d'un item au panier
    }

    public void printTitles(){
        //code d'affichage du titre de chaque livre
    }

    public BooksBag addItems(Bag b) {
        //Code d'ajout des items au panier

        return this;
    }
}

//Deuxième implantation du panier.
public class FlowersBag implements Bag{
```

```
    public void addItem(Item i) {
    }

    public void compterNombreDePetale (){
        //Code permettant de compter le nombre de pétale
    }

    public FlowersBag addItems(Bag b) {
        //Code d'ajout au panier

       return this;
    }
}
```

Dans cette première version, nous n'utilisons pas le principe de covariance. la méthode impactée est addItems :Bag->Bag.

Selon la sous-classe, cette méthode aura un type retour différent. Les méthodes sont bien redéfinies et non surchargée.

```
public class Main {
        public static void main(String[] args) {
                BooksBag bbAutres = new BooksBag();
                FlowersBag fbAutres = new FlowersBag();

                BooksBag bb = new BooksBag();(1)
                FlowersBag fb = new FlowersBag();(2)

                BooksBag bag = bb.addItems(bbAutres); (3)
                FlowersBag fbag = fb.addItems(fbAutres);
(4)

                FlowersBag bag = bb.addItems(bbAutres);
(5)
                BooksBag fbag = fb.addItems(fbAutres); (6)

                FlowersBag bag1 =
(FlowersBag)bb.addItems(bbAutres); (7)
                BooksBag fbag2 =
(BooksBag)fb.addItems(fbAutres); (8)
        }
}
```

Dans le code ci-dessus nous n'avons plus recours au mécanisme de cast. Les erreurs de typage des lignes (5) (6) (7) (8) seront détectées à la compilation et à l'exécution.

Nous avons maintenant un code plus cohérent et plus robuste. Cependant il reste une faille au niveau des arguments. Le principe de la contravariance n'est pas encore implanté dans le langage Java.

Brièvement : ce principe permet de spécialiser les arguments d'une méthode sur le même principe que la covariance.

6. La covariance et le JDK 5 (et versions ultérieurs)

La covariance n'est pas utilisée dans le cadre du JDK 5 sur les classes historiques.

On utilise bien souvent les interfaces
pour laisser des points d'entrer aux développeurs souhaitant étendre les fonctionnalités de l'API.

Par exemple, l'interface List est implémentée de plusieurs façons. Ici, nous retiendrons deux classes l'ArrayList et la LinkedList. Chacune implante une méthode de type clone : Object (méthode issue de l'interface Cloneable).

Dans un contexte autre que celui d'une API (ici le JDK), il aurait été judicieux de remplacer le type retour Object par ArrayList dans un cas et LinkedList dans l'autre. Ceci afin de pouvoir manipuler le bon type sans avoir à caster l'objet retourné.

Hors c'est bien le type Object qui a été utilisé. Lors de l'introduction de la covariance à partir
de la version 5 du JDK, les développeurs de Sun on décider de ne pas modifier les classes existantes. Ceci surement dans un soucis de compatibilité ascendante. Néammoins, ils auraient pu créer de nouvelles classes ou adapter les anciennes comme ils l'ont fait pour les Génerics.

7. Conclusion

Le mécanisme de la covariance apporte une nouvelle perspective au sein du langage Java. Comme les Generics, ce principe donne la possibilité de détecter les erreurs de typage plus tôt durant le développement. Cela permet de diminuer les erreurs au moment de la liaison tardive.

J'espère que cet article vous donnera un bon aperçu du principe de la covariance de type en Java.

Retrouvez l'article de Fabrice Sznajderman en ligne : Lien1

Liste des plugins Eclipse pour installer un environnement de développement complet

Ce document présente une liste des distributions et des plugins Eclipse qui, je l'espère, est assez à jour et complète. Elle a pour but de permettre aux développeurs de trouver rapidement les plugins qui leur seront utiles en fonction de leurs besoins.

1. Les différentes distributions

1.1. Les distributions officielles

Elles sont disponibles à l'adresse suivante : Lien2. Elles permettent d'avoir un environnement avec des plugins pré-installés en fonction de l'utilisation qu'on veut en faire. Bien évidemment, il est possible de rajouter/supprimer des plugins sur chacune de ces distributions :

- Eclipse Classic
- Eclipse IDE for java developers
- Eclipse IDE for java EE developers
- Eclipse IDE for C/C++ developers
- Eclipse for RCP/Plug-in developers

Les noms de ces distributions étant explicites, je ne détaille pas leurs spécificités. On trouve également dans les projets de plugins Eclipse, des versions « All-in-one », incluant Eclipse, le plugin souhaité ainsi que ces dépendances. On a par exemple :

- « WTP All-in-one » (http://www.eclipse.org/webtools/), qui inclut la version complète de WTP et quelques autres plugins pour le développement J2EE
- « TPTP All-in-one » (http://www.eclipse.org/tptp/), qui inclut les plugins TPTP et toutes ses dépendances (BIRT, GEF, etc.)

1.2. EasyEclipse

On trouve ces distributions sur le site Lien3. Elles sont distribuées en OpenSource, et fonctionnent de la même manière que les

distributions officielles : elles intègrent un certain nombre de plugins pour offrir des versions orientées en fonction de l'utilisation qu'on en a :

- EasyEclipse Expert Java
- EasyEclipse Desktop Java – développement de clients riches avec une interface
- graphique Swing ou AWT
- EasyEclipse Server Java
- EasyEclipse Mobile Java – développement d'application pour clients mobiles (J2ME)
- EasyEclipse Plugin Warrior – développement de plugins Eclipse
- EasyEclipse for LAMP – développement en PHP, Python, Perl, et Ruby
- EasyEclipse for PHP
- EasyEclipse for Ruby and Rails
- EasyEclipse for Python
- EasyEclipse for C/C++

1.3. MyEclipse

Contrairement à tout ce qui précède, les distributions MyEclipse ne sont pas disponibles gratuitement. Il en existe deux : une version standard et une version professionnelle, que l'on peut trouver à l'adresse Lien4. Ces distributions intègrent de nombreux plugins pour le développement J2EE (UML, Struts, EJBs, Ajax, explorateurs de bases de données...), dont la liste complète est disponible sur leur site.
Cette distribution étant payante, je n'ai pas d'information sur sa qualité par rapport aux plugins OpenSource qu'on peut trouver dans la communauté Eclipse.

1.4. Autres distributions

- Obeo (Lien5) – Une version d'Eclipse orientée vers la modélisation avec la suite de plugins Acceleo
- Eclipse Discovery (Lien6) – Génération à la volée d'une archive Eclipse incluant les plugins qu'on a sélectionnés
- Amateras Eclipse HTML Editor (Lien7) Distribution de
- Eclipse Callipso fournie avec les plugins Amateras pré-installés (HTML Editor, Struts IDE, Faces IDE, Amateras UML & ERD, etc.)
- Flex Builder IDE (Lien8) – Environnement dedéveloppement de clients riches d'Adobe basé sur Eclipse (payant)

Retrouvez tous les plugins Eclipse proposés par Benoît Courtine en ligne : Lien9

Les livres Java

Sun Certified Programmer for Java 5

100% complete coverage of all official objectives for Sun Java exam 310-055.
Exam Objective Highlights in every chapter point out exam objectives to ensure you're focused on passing the examReal-world exercises--Step-by-step instruction modeled after the hands-on exam questions.
Exam Watch sections in every chapter highlight key exam topics covered.
Simulated exam questions match the format, tone, topics, and difficulty of the real exam.

Covers all exam 310-055 topics, including:

- Declarations and Access Control
- Object Orientation
- Assignment and Initialization
- Operators
- Flow Control, Exceptions, and Assertions
- I/O, Formatting, and Parsing
- Generics
- Collections
- Inner Classes
- Threads
- Java Development
- The best fully integrated study system available.
- CD-ROM includes:

Complete MasterExam practice testing engine, featuring:

- One full practice exam
- Detailed answers with explanations
- Score Report performance assessment tool
- Electronic book for studying on the go
- Bonus downloadable MasterExam practice test (with free online registration)

Critique du livre par la rédaction (Gildas Cuisinier)

Avant tout, je me permettre de bien remettre ce livre dans son contexte : C'est un guide pour la certification SCJP !
Donc si c'est un livre pour apprendre le language Java que vous cherchez, passez votre chemin.

Cela étant dit, ce livre très utile et très intéressant, les deux auteurs du livre sont les co-créateurs de l'examen et donc connaissent très bien le sujet traité par le livre
Fréquemment ils mettent en avant ce qui est important de savoir, tout comme ils informent le lecteur lorsqu'un sujet est hors propos pour la certification.

Le livre en lui même est très agréable à lire, malgré des sujets parfois difficiles à appréhender. Personnellement j'ai dévoré ce livre une première fois, pour le relire par après plus en détail.

L'organisation du livre quant à elle est très clair. Un chapitre reprend un sujet important pour le décortiquer en détail.
Et pour chacun chaque section, un lien est fait vers le ou les objectifs de la certification que celle-ci couvrera.
En fin de chapitre est présent un "Two-Minute Drill", un rappel très succint des informations traités par le chapitre, et un mini test de tester vos acquis sur les sujets du chapitre.

En fin de livre, un examen blanc complet est présent, celui-ci contient des questions du même style que celles qui seront posées durant l'examen.
Enfin, un examen supplémentaire est disponible sur le CD-Rom accompagnant le livre, ainsi qu'un autre en téléchargement sur le site du livre.

En conclusion, je dirai que pour ma part ce livre à répondu à son objectif, c'est grâce à lui que j'ai réussi mon examen SCJP 5.

Retrouvez ce livre sur la rubrique Java : Lien10

Les Cahiers du programmeur Java EE 5

Ce cahier détaille la conception d'un site de commerce électronique avec UML et Java EE 5. Inspirée du Java Petstore,

l'étude de cas se construit au fil des chapitres en appliquant les spécifications Java EE 5 : EJB 3.0, JPA 1.0, Servlet 2.5, JSP 2.1, JSF 1.2, Web Services 1.2, JAXB 2.0, JAX-WS 2.0, JavaMail 1.4, JMS 1.1. L'application est déployée dans le serveur GlassFish et utilise la base de données Derby.

Cet ouvrage s'adresse aux architectes et développeurs confirmés qui veulent découvrir les nouveautés de Java EE 5 ou migrer leurs applications J2EE 1.4 existantes. Il montre comment s'imbriquent les différentes API de Java EE 5 dans une application internet-intranet.

Critique du livre par la rédaction (Yann D'Isanto)

Bien que des notions élémentaires de la plateforme Java EE soient suffisantes pour aborder ce livre, il requiert toutefois d'avoir de solides connaissances de la plateforme Java SE ainsi que de bons acquis en UML et XML.

Cet ouvrage tient toutes ses promesses. Il réussit le tour de force d'aborder un nombre impressionant d'APIs de la plateforme Java EE de façon claire et concise. Celles-ci ne sont donc pas abordées dans leur intégralité (un livre ne suffirait pas) mais elles le sont suffisamment pour commencer à savoir s'en servir et nous donner l'envie d'aller plus loin. Ces APIs font toutes partie des spécifications constituant Java EE 5 (EJB3, JPA, JSP, JSF, JSTL, JMS, JAX-WS, JAXB, JavaMail, ...). Pour les personnes voulant approfondir ces notions, l'auteur a eu la très bonne idée de donner, tout au long de son ouvrage, des références sur les sujets qu'il aborde.

Le livre est construit autour du développement d'une application web complète (de type site marchand). C'est cette approche qui permet de couvrir de façon concrète le large panel d'APIs abordées.

Autre point intéressant, la première partie est consacrée à la conception de l'application où l'auteur partage son expérience et son savoir faire en expliquant ses choix (attitude qu'il adopte tout au long du livre).

Agréable à lire et au contenu d'une grande qualité, cet ouvrage m'apparaît être un indispensable pour tout développeur Java EE désireux de connaître la plateforme Java EE 5.

Retrouvez ce livre sur la rubrique Java : Lien11

Swing La synthèse

Swing est une librarie Java qui permet de créer et de gérer des interfaces graphiques évoluées.

Le but de cet ouvrage n'est pas de présenter la liste exhaustive des composants Swing, mais d'expliquer les principes de cette bibliothèque de classes et d'en offrir une vue d'ensemble.

Les composants les plus courants sont analysés en détail, amenant l'explication des concepts de layout, d'événement, de drag and drop et d'architecture MVC, ainsi que l'étude des composants plus complexes. Enfin, des problématiques plus avancées sont abordées, telles que le parralélisme, le multifenêtrage, les tests ainsi qu'une ébauche de framework.

La démarche s'appuie sur de nombreux exemples correspondant à des cas d'utilisation dans le monde professionnnel. L'un d'eux est enrichi tout au long de l'ouvrage, au fur et à mesure de la présentation de ces concepts.

Cet ouvrage s'addresse donc à des développeurs devant programmer une interface graphique avec Swing, à des architectes devant concevoir un système où Swing intervient ou encore à des chefs de projet ou des décideurs ayant à comprendre les principes globaux de Swing. Tous pourront acquérir le recul nécessaire aussi bien lors du développement que lors de choix techniques ou architecturaux.

Critique du livre par la rédaction (Baptiste Wicht)

Ce livre est destiné à toutes les personnes utilisant Swing. Il va vous expliquer les principaux concepts de Swing. Vous allez les apprendre via une série d'exemples ludiques. A la fin de chaque chapitre, vous allez pouvoir appliquer les différentes notions à une petite application.

On va commencer par voir les concepts fondamentaux de Swing. C'est-à-dire la notion de composants, de container, les fenêtres ainsi que les looks and feels. On va continuer ensuite avec la gestion des événements. Puis on va partir sur les composants plus complexes comme la JList, la JTable et le JTree. On va ainsi voir la notion de modèles, de renderers ainsi que d'editors. Sans oublier l'architecture MVC de Swing. Après avoir découvert toutes ces notions, un chapitre entier sera consacré aux composants texte avec notamment la gestion des Document, des Action ainsi que de l'undo-redo. Puis un autre chapitre pour le drag and drop. Vous y découvrirez 2 manières de faire du drag-and-drop, la deuxième étant seulement disponible depuis le JDK 4.

Après nous avoir parlé de tous les concepts de "base" de Swing, l'auteur va nous expliquer le concept de threading avec Swing. Un concept très important mais souvent mal compris par les débutants et même par les utilisateurs plus expérimentés. Ce concept sera expliqué avec une série de tests pour montrer la différence de performances de Swing avec ou sans respect de l'Event Dispatching Thread. On va également apprendre quels sont les nouveautés qui ont vu le jour dans les dernières versions de Java (1.4 et 5.0 pour ce livre). Puis vient le dernier chapitre sur le formalisme UML, mais je ne trouve pas que ce chapitre apporte grand-chose au livre vu la faible complexité des diagrammes UML utilisés.

En conclusion, ce livre va vous apprendre les principaux concepts de Swing et d'une très bonne manière. Après la lecture de ce livre, vous devriez parfaitement maîtriser les bases de Swing et vous devriez pouvoir vous lancer dans le développement d'interfaces graphiques en Swing. Néanmoins, ce livre n'est pas un catalogue Swing montrant tous les composants et leurs fonctions, la Javadoc et les tutoriaux de Sun étant nettement suffisants à ce niveau-là. On peut néanmoins regretter le manque de clarté de certains chapitres, comme celui sur les Javabeans ou encore la brièveté du chapitre sur l'undo-redo ou celui sur la deuxième manière de faire du drag-and-drop. Mais mis à part ces petits défauts, ce livre reste une excellente introduction au framework Swing et je ne peux que vous le conseiller.

Retrouvez ce livre sur la rubrique Java : Lien12

Comment créer ses pages JSP ?

File -> New File -> Web -> JSP
Dans les options, indiquer si c'est un fichier JSP (par défaut, extension .JSP), ou un document JSP (syntaxe XML, extension .JSPX). N'oubliez pas de cocher la case Create as a JSP Fragment, si c'est juste un fragment de page (à inclure dans une page JSP ou un document JSP).

Notez que depuis NetBeans 5.0, vous avez également une palette d'éléments, ce qui vous permet de facilement rajouter des formulaires, des tableaux à vos pages JSP.

Comment créer une application Web à l'aide de JSF ?

SNetBeans fournit un support light pour JSF. Lorsque vous créez un projet pour une application Web, vous pouvez, après avoir indiqué à la deuxième étape son emplacement, le serveur sur lequel il tournera, indiquer le framework que vous désirez intégrer dans votre application web.

Un des frameworks proposés est JSF 1.1. (l'autre étant Struts 1.2.7).

Une fois le framework JSF 1.1 sélectionné, vous devez indiquer si l'EDI NetBeans doit valider le XML et/ou vérifier les objets.

L'EDI NetBeans aura alors créé le fichier face-config.xml et configuré le fichier web.xml selon vos désidératas.

Lorsque vous éditez le fichier faces-config.xml, vous pouvez cliquez-droit et sélectionner l'entrée Java Server Faces pour vous voir proposer les actions suivantes:

- Add Navigation Rule: Vous permet de définir des règles de navigation
- Add Navigation Case: Vous permet de définir la navigation entre vos pages.
- Add Managed Bean: pour définir les Java beans qui pourront être pris en charge par JSF.

Comment créer une application Web à l'aide de Struts ?

NetBeans fournit un support light pour Struts. Lorsque vous créez un projet pour une application Web, vous pouvez, après avoir indiqué à la deuxième étape son emplacement, le serveur sur lequel il tournera, indiquer le framework que vous désirez intégrer dans votre application web.

Un des frameworks proposés est Struts 1.2.7 (l'autre étant JSF 1.1).

Une fois le framework Struts 1.2.7 sélectionné, vous devez valider l'URL Pattern qui vous est proposé. N'oubliez pas de cocher la case Add Struts TLD si vous désirez utiliser les TLD de Struts dans vos page JSP.

L'EDI NetBeans aura alors créé les fichiers struts-config.xml, validation.xml, validator-rules.xml, tile-defs.xml, et configuré le fichier web.xml selon vos désidératas.

Lorsque vous éditez le fichier struts-config.xml, vous pouvez cliquer-droit et sélectionner l'entrée Struts pour vous voir proposer les actions suivantes:

- Add Action
- Add Forward/Include Action:
- Add Forward
- Add Exception
- Add ActionForm Bean
- Add ActionForm Bean Property

Y a-t-il un support pour Tomcat ?

Depuis que je connais NetBeans (version 3.1), il a toujours intégrer une version de Tomcat. L'EDI NetBeans 5.0 est livré avec la version 5.5.9 de Tomcat. Vous n'avez rien de spécial à configurer pour exécuter votre application sous Tomcat.

Comment déboguer un projet Tomcat ?

Rien de plus simple pour cela.
Il vous suffit de sélectionner, dans la fenêtre Projets, le noeud de la page JSP ou HTML de votre projet Tomcat, de cliquez-droit dessus et de sélectionnez debug. Tomcat sera alors automatiquement lancé en mode debug. Et vous pourrez déboguer votre application comme vous le feriez pour une application Java. Sachez que NetBeans supporte le débogage pas à pas également dans les pages JSP.

Retrouvez ces questions et de nombreuses autres sur la FAQ Netbeans : Lien12

Commencer à développer avec le framework symfony

Ce tutoriel vous donnera les bases pour bien comprendre le fonctionnement global du framework PHP symfony. Il s'adresse principalement à des développeurs PHP ayant des notions en programmation orienté-objet.

1. Introduction

Le web est un environnement est en pleine mutation. De nombreuses technologies et outils font leur apparition et nous proposent tous les jours de plus en plus d'interactivités dans nos pages web (« Atlas », script.aculo.us, Adobe Integrated Runtime…). Toutes ces technologies transforment nos chers navigateurs en véritable plateformes. Cependant, on oublie souvent en voyant toutes ces technologies que l'évolution se situe aussi du côté du serveur. Le langage PHP lui aussi témoigne de ces profonds changements dans notre manière de développer. Le langage PHP dispose depuis quelques temps de bibliothèques de code permettant d'optimiser le développement avec ce langage (tel que PEAR). Cependant, l'avènement de Ruby on rails a démontré les nombreux avantages à posséder une architecture MVC. symfony fait partie des frameworks inspirés du modèle proposé par rails (tels que Prado, CodeIgniter, ou bien CakePHP)

symfony n'est pas développé de zéro. Il intègre d'autres composants déjà approuvés par un grand nombre tel que Propel pour l'ORM, une couche d'abstraction de base de données propulsée par Creole, et Mojavi pour l'architecture MVC.

La documentation sur ce framework est très aboutie, cependant, certains concepts peuvent être longs à acquérir. Ce tutoriel a pour objectif de vous donner des bases solides dans la compréhension de ce framework afin que vous puissiez commencer à développer avec, ou continuer votre apprentissage.

2. Avant de commencer

Lorsque l'on est développeur PHP, on a souvent l'habitude d'utiliser des scripts, des bibliothèques déjà existantes dans nos projets (et les includes sont légions en en-tête de nos scripts). A la différence d'autres frameworks PHP – tel que le Zend Framework – ou de librairies tierces, symfony nécessite une installation un peu plus lourde.

symfony est utilisable de deux façons :

- **Traditionnelle**: Le framework est installé directement sur le serveur web dans le répertoire de PHP. Il est partagé entre tous les projets symfony du serveur. L'utilisation des liens symboliques et la gestion des droits unix est nécessaire.
- **Freeze**: Dans ce mode, le dossier de votre projet embarque tous les fichiers nécessaires au framework. On ne travaille généralement pas avec ce mode, par contre, il est tout à fait adapté au déploiement.

Vous avez la possibilité de passer d'une application traditionnelle à une application freeze et vice-versa.

2.1. L'installation de symfony

Il y a de nombreuses façons d'installer symfony, selon vos connaissances, votre plateforme, … La plus simple reste l'utilisation de PEAR. Voici un ensemble de liens qui vous permettront d'installer symfony quelque soit votre plateforme :

- Installer symfony sur Ubuntu Dapper Drake : Lien14
- Installer symfony via PEAR : Lien15
- Installer symfony sous MacOsX : Lien16
- Installer symfony sur WAMP : Lien17

2.2. Développeurs PHP ? Oubliez vos habitudes !

Les détracteurs du langage PHP ont de nombreuses raisons de se plaindre : lenteur, permissivité, pas de séparation entre le code et le design, etc… symfony apporte de nombreuses réponses (que vous découvrirez tout au long de ce tutorial ;-). Cependant, tout a un prix ! Certains développeurs PHP « traditionnels » peuvent être confrontés à des principes ou à l'utilisation d'outils auxquels ils ne sont pas habitués.

2.2.1. Vive la console !

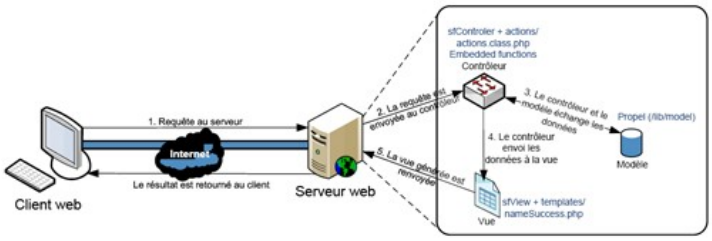
Développer avec symfony, c'est aussi réveiller le côté vim qui est en vous ! En effet, vous serez amené à exécuter des lignes de commandes de façon très régulière. Si cela fait un moment que vous n'avez pas touché à la ligne de commande, je vous conseille de faire un tour par la FAQ Linux (Lien18). Ces commandes permettent notamment de créer un nouveau projet, une nouvelle application, d'utiliser Propel… Vous trouverez sur le blog d'Andréia Bohner une fiche récapitulative (Lien19) des commandes symfony les plus importantes. Avant de faire une commande, assurez-vous d'être à la racine de votre projet !

2.2.2. Le design pattern MVC et l'objet

symfony est entièrement basé sur le design pattern MVC (Modèle-Vue- Contrôleur). Utilisé dans d'autres d'autres technologies (notamment Java), il n'a été largement diffusé dans le monde PHP qu'avec des frameworks tels que symfony. Le MVC est un pattern architectural qui sépare les données (c-à-d le modèle), l'interface homme-machine (la vue) et la logique de contrôle (le contrôleur).

Ce modèle de conception impose donc une séparation en 3 couches : - Le modèle : Il représente les données de l'application. Il définit aussi l'interaction avec la base de données et le traitement de ces données. - La vue : Elle représente l'interface utilisateur, ce avec quoi il interagit. Elle n'effectue aucun traitement, elle se contente simplement d'afficher les données que lui fournit le modèle. Il peut tout à fait y avoir plusieurs vues qui présentent les données d'un même modèle. - Le contrôleur : Il gère l'interface entre le modèle et le client. Il va interpréter la requête de ce dernier pour lui envoyer la vue correspondante. Il effectue la synchronisation entre le modèle et les vues. symfony

implémente donc trois couches répondant à ce pattern. Le schéma ci-dessous illustre le mécanisme du MVC dans un projet symfony lorsque l'utilisateur navigue sur le site.



Pour une courte introduction à MVC, je vous conseille la lecture de la première partie du tutoriel de Baptiste Wicht (Lien20) (pour une introduction rapide), et la lecture du cours de Serge Tahé (Lien21) pour aller plus loin.

La programmation avec le framework symfony est clairement orienté objet. Cependant de nombreux développeurs PHP n'ont pas encore sauté le pas. Si vous êtes dans ce cas, il vous faudra donc passer par l'étape apprentissage de l'objet ! (par exemple en consultant les tutoriels disponibles à ce sujet). On peut maintenant entrer dans le vif du sujet : la création de votre premier projet !

3. La création d'un projet

La création d'un projet symfony commence par la création de la structure de base de celui-ci, c'est-à-dire un ensemble de dossiers et de fichiers.

3.1. Projets, Applications et cie

symfony permet à partir d'un seul projet de gérer plusieurs applications, par exemple une pour le frontend et une autre pour le backend. Voici quelques précisions sur les différents niveaux d'un projet :

- **Le projet** : C'est la structure qui englobe tout. C'est à son niveau que l'on spécifie les informations sur la base de données.
- **L'application** : Elle correspond généralement à un site. C'est à ce niveau qu'il y a le plus de configuration à effectuer. Un projet peut comporter plusieurs applications, cependant, dans la plupart du cas, il n'y en aura qu'une.
- **Les modules** : Ce sont des 'unités fonctionnelles' de votre applications. (Par exemple : clients, produits, commandes). C'est à ce niveau que vous allez travailler le plus !
- **Les actions et les templates** : Les modules comportent des couples d'actions-templates. C'est l'application du pattern MVC, et également la séparation entre la partie « traitement » et la partie « affichage ». Toutes les opérations sur la base de données et les autres traitements (web services, ftp, mail, etc…) sont réalisés dans le fichier d'actions (monModule/actions/action.class.php), c'est ensuite le fichier de template (monModule/templates/monActionSuccess.php) qui se charge du rendu graphique.

3.2. La création d'un nouveau projet

La création d'un nouveau projet passe par l'utilisation de la ligne de commande. Il vous faut créer un répertoire dans lequel placer votre projet. Vous pouvez le faire via l'interface graphique (selon votre serveur) ou via la ligne de commande :

```
mkdir monProjet
```

```
cd monProjet
```

Une fois que vous êtes bien dans le répertoire de votre projet, exécuter la commande de création d'un nouveau projet, en remplaçant monProjet par le nom de votre projet :

```
symfony init-project monProjet
```

3.3. La création d'une application

Vous devez ensuite créer une application dans votre projet (vous pourrez par la suite refaire cette étape si vous souhaitez créer plusieurs applications au sein de votre projet). Là aussi, une simple ligne de commande suffit :

```
symfony init-app monProjet monAppli
```

Une nouvelle arborescence va être créé dans monProjet/apps/monAppli. Elle contient un ensemble de répertoires :

- **Config** : Ce répertoire contient tous les fichiers de configuration propre à l'application (cache, routing, sécurité, ...)
- **I18n** : C'est ici que sont stockés les fichiers pour l'internationalisation (régionalisation) de votre projet, notamment les fichiers de traduction (généralement messages.fr.xml).
- **Lib** : Ce dossier contient les librairies personnelles que vous pouvez ajouter. Il contient également un fichier appelé myUser.class.php. **Ce fichier ne doit pas être supprimé ni renommé** (il dérive d'une classe User nécessaire au fonctionnement du site, y compris si votre site ne comporte pas de gestion d'utilisateur !). Vous pouvez surcharger les méthodes afin de personnaliser la gestion des utilisateurs.
- **Modules** : Vide lors de la création de votre application, c'est lui qui contiendra tous les modules de votre application (notamment les fichiers d'actions et les templates).
- **Template** : Contient le template dynamique par défaut (layout.php). C'est lui qui inclut tous les metas et le titre de la page. C'est également lui qui réalise l'inclusion du template de l'action en cours.

3.4. Finitions

Il vous reste une dernière petite chose à effectuer avant de profiter de symfony. Vous devez créer un lien symbolique entre un répertoire de l'installation de symfony et votre projet, afin d'avoir les styles par défaut de symfony et l'accès à la barre web de débogage. Pour se faire, vous devez effectuer la commande suivante (c'est un exemple pour la machine virtuelle, il vous faudra l'adapter à votre version de *nix et votre installation de symfony)

```
ln -s /usr/share/php/data/symfony/web/sf ./web/sf
```

Vous pouvez maintenant passer à la configuration de votre projet !

4. Les fichiers de configuration

Les fichiers de configuration au sein de symfony sont nombreux. Cette partie va présenter où ils se situent, leur fonctionnement ainsi que leur syntaxe, et enfin quelques informations sur les principaux fichiers de configuration à connaître et à modifier.

4.1. La configuration dans symfony

La configuration s'effectue à plusieurs niveaux : Le projet, l'application et les modules. On peut tout a fait définir une configuration globale à une application mais surcharger la configuration d'un module particulier en changeant les informations de son fichier de configuration (les noms et le contenu sont identiques entre les fichiers de configuration d'une application et d'un module). Les fichiers de configuration pour une application sont situés dans monProjet/apps/monAppli/config. Les fichiers de configuration pour un module sont situés dans monProjet/apps/monAppli/modules/monModule/config. Il n'y a pas de documentation exhaustive sur ces fichiers de configuration. Cependant la plupart des informations utiles sont situées directement dans les fichiers de configuration, sous forme de commentaires.

4.2. Syntaxe des fichiers YAML

Les fichiers de configuration de symfony sont quasiment tous au format YAML (extension .yml). Basé sur un fichier texte, ils présentent une syntaxe simple. Les différents niveaux hiérarchiques sont représentés par des espaces avant le texte.

```
prod:                                <- Niveau 1: la
ligne se termine par deux points (:)
 .settings:                  <- Niveau 2: la ligne
commence par des espaces et se termine par deux points
  no_script_name: off <- Paramètre: la valeur commence
au premier caractère (espace non compté) après les deux
points (:)

dev:                                 <- Niveau 1
 error_reporting: 4095       <- Paramètre
  web_debug: on              <- Paramètre
  cache: off                 <- Paramètre
```

Les fichiers YAML ne supportent pas les tabulations. N'utilisez que des espaces !

4.3. Les principaux fichiers de configuration

Il y a de nombreux fichiers de configuration dans symfony. Voici la liste des principaux fichiers, ainsi que les points importants pour chacun des fichiers.

4.3.1. config.php

Ce fichier est généré automatiquememnt à la création de l'application. C'est un fichier PHP. Il contient les emplacements du framework. Ce fichier ne doit être modifié que si vous changez l'emplacement de l'installation du framework. Il n'est pas surchargable (au niveau des modules).

4.3.2. i18n.yml

Ce fichier contient les paramètres d'internationalisation de l'application. Il permet de paramétrer la langue par défaut, le 'backend' (c'est-à-dire le type de fichiers permettant la traduction. Il y a, entre autres, le format XLIFF, PO et CSV).

4.3.3. logging.yml

symfony embarque un système de logs. C'est ce fichier qui permet de le configurer. Vous pouvez notamment choisir le niveau de log, la période, s'il est rotatif ou pas, l'emplacement du fichier de log.

4.3.4. routing.yml

Ce fichier permet de gérer la réécriture d'URL. Il est important de le modifier afin que la première page de votre projet ne soit pas la page par défaut de symfony. Pour faire ceci, vous devez créer une action dans un module qui sera exécutée lors de l'ouverture de votre site.

```
# default rules
homepage:
url: /
param: { module: monModule, action: index }
```

4.3.5. security.yml

symfony intègre un système de contrôle d'accès, vous permettant de gérer par modules et/ou par actions qui a le droit d'executer l'action ou de modifier des données. Le fichier security.yml, qui se retrouve au niveau de l'application (respectivement des modules), vous permet de paramétrer pour chacun des modules (resp. actions) s'il faut être identifié et s'il faut appartenir à un certain groupe d'utilisateurs. Il est préférable d'utiliser le fichier security.yml au niveau des modules et non de l'application. Cependant, il n'existe pas lors de la création du module. Il faut donc le créer manuellement, dans le dossier monProjet/apps/monAppli/modules/monModule/config/. Voici un exemple de fichier :

```
read: # Début de la section pour la lecture (toutes
actions)
is_secure: off # Positionné à off, tout le monde peut
accéder à l'action en lecture
update: # Début de la section pour la modification
is_secure: on # Positionné à on, il faut être
"connecté" pour activer la mise a jour
credentials: admin # Ici, il faut avoir le privilège
'admin' pour réaliser une maj
editArticle: # Début de la section pour l'action
editArticle
credentials: admin # Il faut avoir le privilège 'admin'
pour pouvoir executer l'action
editArticle
publishArticle: # Début de la section pour l'action
publishArticle
credentials: publisher # Il faut avoir le privilège
'publisher' pour pouvoir executer l'action
publishArticle
```

4.3.6. settings.yml

C'est dans le fichier settings.yml que sont définis les principaux paramètres d'une application symfony : internationalisation, cache, desactivation de l'application, utilisation de composants, gestion de l'encodage, … Il est disponible uniquement au niveau de l'application. Vous avez la possibilité de définir plusieurs environnements dans de fichier. Ainsi, vous pouvez avoir différents paramètres pour le développement (par exemple, toutes les options de débogage activées et pas de cache), pour les tests (cache+débogage) et pour la production (juste le cache).

```
prod:
  .settings:
    web_debug:              off
    cache:                  on

dev:
  .settings:
    web_debug:              on
    cache:                  off

test:
  .settings:
    cache:                  on
```

```
    web_debug:              on
```

4.3.7. view.yml

Vous commencez votre site sous symfony. En bon développeur, vous commencez à insérer dans votre page les liens vers vos fichiers CSS et Javascript ? Oubliez cela ! C'est le fichier view.yml qui va gérer cela pour vous ! La mise en page est réalisée à partir des éléments suivants :

- Le « coeur » de la page est généré par le fichier de template
- La mise en page globale (l'en-tête, le pied de page, éventuellement les colonnes) proviennent du fichier de layout (layout.php)
- Tout le reste - c'est-à-dire le titre de la page, les fichiers CSS et Javascript à inclure, les métas, les mots clés, la description, le délai du cache... - est configuré dans le fichier view.yml.

Vous avez la possibilité de définir des paramètres globaux dans le fichier view.yml de l'application (monProjet/apps/monAppli/config/view.yml), et personnaliser la vue pour certains modules (en créant un fichier view.yml dans monProjet/apps/monAppli/modules/monModule/config/). Cela peut vous être utile, par exemple, si vous utilisez de l'ajax dans un module spécifique.

```
default:
  http_metas:
    content-type: text/html; charset=utf-8

  metas:
    title:        Bienvenue sur developpez.com
    robots:       index, follow
    description:  developpez.com le site no 1 des
developpeurs
    keywords:     informatique, developpement, cours,
tutorial, programmation
    language:     fr

  stylesheets:    [main,develz]

  javascripts:    [jquery/src/jquery/jquery.js,
libperso/popup.js]

  has_layout:     on
  layout:         layout
```

Exemple de fichier view.yml au niveau d'un module

```
articleSuccess:
  metas:
    title: Article - Developpez.com

faqSuccess:
  metas:
    title: Foire aux questions - Developpez.com
  javascripts:        [qr.js]
```

Maintenant que les fichiers YAML n'ont plus de secrets pour vous, nous allons passer à la partie modèle, c'est-à-dire la base de données.

5. La base de données

symfony intègre de nombreux composants pour la gestion des bases de données. Il est capable de gérer la connexion à différents moteurs, créer les tables, importer des données… Cette partie va vous permettre d'avoir les bases de la gestion des bases de données sous symfony.

Attention : symfony effectue une conversion implicite entre la version relationnelle et objet d'une base. Le champ nom_utilisateur dans la base de données sera converti en nomUtilisateur (Camel style). Il est fortement conseillé de ne pas utiliser ce style directement dans la base de données.

5.1. Relier son projet à une base de données

Pour relier votre projet à une base de données, vous devez modifier certains fichiers de configuration.

5.1.1. properties.ini

Ce fichier est simple, mais il est très important. C'est lui qui contient le nom du projet. Il doit être identique dans tous les autres fichiers de configuration (databases.yml, Propel.ini, schema.yml).

```
[symfony]
name=monProjet
```

5.1.2. databases.yml

Ce fichier contient les paramètres de connexion à la base de données. Il se situe au niveau du projet (monProjet/config/databases.yml), mais il est possible de le surcharger pour une application (en le créant dans monProjet/apps/monAppli/config/databases.yml)

```
all:
monProjet:
class: sfPropelDatabase
param:
datasource: monProjet
phptype: mysql
hostspec: localhost
database: maBase
username: root
password: toor
```

5.1.3. Propel.ini

Le fichier Propel.ini reprend les mêmes informations que le fichier databases.yml, mais il est destiné à la couche ORM (Object-relational mapping) de symfony.

```
Propel.project = monProjet
Propel.database = mysql
Propel.database.createUrl = mysql://localhost/
Propel.database.url = mysql://locahost/monProjet_dev
```

5.2. Le fichier schema.yml

Le fichier schema.yml contient une représentation du modèle relationnel de la base de données. Il est nécessaire pour pouvoir réaliser le mapping objet, ainsi que pour la génération de pages (CRUD et admin)

Le fichier schema.yml étant un fichier YAML, vous en connaissez déjà la syntaxe. Le niveau 1 correspond à une table et le niveau 2 à un champ. Le contenu du niveau 2 est entre accolades et il regroupe toutes les caractéristiques du champ (type, taille, …). L'exemple ci-dessous vous explique comment structurer votre fichier pour une table.

A partir d'une base de données existante

symfony offre également la possibilité de créer ce fichier (au format YAML ou XML) à partir d'une base de données existante.

Il faut tout d'abord paramétrer la connexion à la base de données (fichier properties.ini, databases.yml et Propel.ini), puis exécuter la commande suivante :

Création du fichier schema au format YAML
```
symfony propel-build-model
```

Création du fichier schema au format XML
```
symfony propel-build-model xml
```

5.3. Préparer symfony pour la base de données

symfony sait maintenant comment est structurée votre base de données. Il faut maintenant faire la passerelle entre la base (relationnelle) et le php (en objet). C'est la génération du modèle. Cette étape va créer les classes correspondantes aux tables indiquées dans le fichier schema.yml. Les fichiers sont générés dans le répertoire monProjet/lib/model/om/. Cette génération est réalisée avec la ligne de commande suivante : symfony propel-build-model. Vous avez la possibilité de surcharger toutes les méthodes créées par symfony dans le fichier maTable.php situé dans monProjet/lib/model/om/. Il ne vous reste plus qu'a créer les tables dans votre base de données. Vous avez deux solutions pour cela, en ligne de commande : Créer un fichier SQL et l'exécuter vous-même :

Création du fichier SQL
```
symfony propel-build-sql
```

Laisser symfony se connecter à la base de données et créer les tables :

symfony crée la base de données
```
symfony propel-build-db
```

6. Les modules

6.1. Création d'un module

La création d'un module s'effectue en ligne de commande. Vous pouvez créer un module vierge (ce sera alors à vous de créer toutes les actions), ou bien de créer un module permettant d'effectuer les opérations de bases sur une table.

6.1.1. Création d'un module vierge

La création d'un module vierge se fait via la ligne de commande :

```
symfony init-module monProjet monModule
```

Il est conseillé de ne pas utiliser de majuscules ni de caractères spéciaux ou d'espaces dans le nom du module. Ce module nouvellement créé ne contient que le fichier actions.class.php et l'arborescence d'un module. Si votre module n'utilise pas ou très peu la base de données, c'est un module vierge qu'il faut créer. Cependant, si vous créez un module qui utilise la base de données, vous avez intérêt à utiliser un module CRUD.

6.1.2. Création d'un module CRUD

L'un des grands intérêts de symfony est de pouvoir créer des modules vous permettant d'effectuer toutes les opérations courantes sur votre base de données, à savoir :

- Voir la liste des enregistrements
- Voir un enregistrement
- Modifier un enregistrement
- En créer un nouveau
- En supprimer

C'est la génération CRUD (Create-Retrieve-Update-Delete). La génération s'effectue par la ligne de commande suivante :

```
symfony propel-generate-crud monAppli monModule
Maclasse
```

Le paramètre monModule correspond au nom du module que vous souhaitez créer. Maclasse correspond au nom de la classe de la base de données. Les deux paramètres sont généralement identiques, mais le second commence toujours par une majuscule. Cette génération va créer toutes les actions et les templates requis pour effectuer ces opérations de base. Vous pouvez bien sûr modifier ce module comme le module vierge.

6.2. Structure des modules

Lors de la création d'un module, un nouveau répertoire est créé dans /monProjet/apps/monAppli/modules. Il contient 5 dossiers :

- Actions : Ce dossier contient le fichier actions.class.php. C'est lui qui contient toutes les méthodes-actions du module
- Config : il permet de surcharger la configuration de l'application (il est vide par défaut). La configuration est principalement surchargée pour la sécurité (security.yml) et pour l'affichage (view.yml)
- Lib : Ce dossier permet d'insérer des classes personnalisées accessibles uniquement dans ce module.
- Templates : Contient tous les templates associés aux actions du module
- Validate : Ce dossier contient les fichiers de configuration permettant la validation automatique des formulaires.

6.2.1. Création d'une nouvelle action

Toutes les actions pour un module sont regroupées dans un fichier : /monProjet/apps/monAppli/modules/monModule/actions/actions.class.php. Tout d'abord, il faut créer dans ce fichier une nouvelle méthode (public function). Le nom de la méthode doit obligatoirement commencer par execute et être suivie du nom de votre action (la première lettre en majuscule) :

```
public function executeNouveauclient(){
…
}
```

Ce fichier va vous permettre d'effectuer vos opérations, mais pas l'affichage. Pour cela, vous devez passer par les templates.

6.3. Les templates

Comme vous avez pu le remarquer, il existe un dossier templates dans votre module. Par défaut, lorsque l'action monAction est exécutée, c'est le fichier monActionSuccess.php dans ce dossier qui est appelé. Vous devez donc créer ce fichier dans le dossier templates. Vous avez également la possibilité de modifier le fichier template qui sera appelé dans le code de votre action. Cela vous permet de modifier dynamiquement le template qui sera appelé par une action donnée.

```
$this->setTemplate('monTemplate');
```

symfony offre également la possibilité de se passer de fichier template et de préparer la sortie directement dans le fichier actions. Cette méthode n'est pas conseillée et vous devez la réserver pour des cas bien spécifiques (comme le renvoi de JSON lors d'un appel ajax).

```
public function executeIndex(){
        echo "<html><body>Bonjour à tous<body></html>";
        return sfView::NONE;
}
```

Ou

```
public function executeIndex(){
        return $this->renderText("<html><body>Hello,
World!</body></html>");
}
```

7. Les objets de base

7.1. Les liens dans symfony

symfony utilise la réécriture d'URL. C'est pourquoi il ne faut pas faire les liens directement en HTML (en utilisant la balise <a link>). Les liens pointent vers une action (login) d'un module (utilisateur) :

```
<?php echo link_to("utilisateur/login") ?>
```

Pour passer des paramètres à la page appelée, vous pouvez utiliser la syntaxe habituelle :

```
<?php echo
link_to("utilisateur/login?email=chris@bulles.org") ?>
```

7.2. Passer une variable de l'action au template

Tous les résultats des traitements que vous réalisez dans votre fichier d'actions doivent être passés au fichier template pour être affichés. Il faut tout d'abord déclarer cette variable dans l'action. Cette variable doit être de la forme : $this->maVar Pour récupérer cette variable dans le template, il suffit de faire : $maVar Ce « changement » de variables entre les actions et les templates peut être déroutant au début. Il est également valable pour « les objets de la base de données » :

dans le fichier actions.class.php

```
$this->utilisateur = UtilisateurPeer ::doSelectOne($cr)
;
```

dans le fichier de template

```
<p>Bienvenue <?php echo $utilisateur->getNom() ?>,
c'est votre <?php echo $utilisateur->getNbConnexion() ?
>eme connexion.</p>
```

7.3. Gérer les utilisateurs et les sessions

symfony intègre un système complet de gestion des sessions et des utilisateurs. Voici les bases de cette gestion, de très nombreuses fonctionnalités ne sont pas abordées ici. La documentation vous permettra d'approfondir ces concepts. Gestion des

7.3.1. Gestion des attributs de la session

Vous avez la possibilité de stocker des attributs dans l'objet session, comme par exemple la liste des articles du panier d'achat. Deux fonctions vous permettent d'écrire et de lire ces attributs :

Dans les actions

```
$this->getUser()->setAttribute('nbArticles', 15) ;
$this->getUser()->getAttribute('nbArticles') ;
```

Dans les templates

```
$sf_user->setAttribute('nbArticles', 15) ;
$sf_user->getAttribute('nbArticles') ;
```

7.3.2. Connexion de l'utilisateur

Vous pouvez stocker l'état de l'utilisateur : connecté ou déconnecté.

Dans les actions

```
$this->getUser()->setAuthenticated(true) ; // Connecte
$this->getUser()->setAuthenticated(false) ; //
Deconnecte
$this->getUser()->isAuthenticated() ; // Vrai si
connecté
```

Dans les templates

```
$sf_user->setAuthenticated(true) ; // Connecte
$sf_user->setAuthenticated(false) ; // Deconnecte
$sf_user->isAuthenticated() ; // Vrai si connecté
```

7.3.3. Gestion des permissions

Même si de nombreux sites se contenteraient d'un état « Administrateur », d'autres ont besoin d'une gestion beaucoup plus fine (un blog peut avoir par exemple un auteur, des utilisateurs pouvant poster des commentaires, et des modérateurs). Voici comment utiliser ces permissions.

Ajouter une permission :

```
$this->getUser()->addCredential('auteur') ;
```

Tester si l'utilisateur a une permission particulière :

```
$this->getUser()->hasCredential('moderateur') ;
```

Supprimer une permission :

```
$this->getUser()->removeCredential('auteur') ;
```

Supprime toutes les permissions :

```
$this->getUser()->clearCredentials() ;
```

7.4. La création d'un formulaire

Vous allez faire votre entrée dans le monde des helpers ! Les helpers sont des méthodes php qui génèrent du code Javascript. Elles permettent de générer du code propre, respectueux des standards et tout ceci simplement. Les helpers sont présents dans de nombreux domaines : Javascript, AJAX, Flash, formatage de données, régionalisation…et les formulaires ! La première étape pour créer un formulaire est de débuter par la balise <form>. Le code suivant va créer l'entête du formulaire

```
<?php echo form_tag('user/create')?>
```

Il faut ensuite créer tous les champs de votre formulaire. Il existe deux grands types d'helpers : les standards (commençant par input) et ceux reliés à la base de données (object helper – commençant par object). Il faut ensuite fermer vous-même le formulaire (en fermant la balise </form>).

7.4.1. Les champs standards

Les extraits de code suivants regroupent tous les helpers standards. C'est une traduction (partielle) du manuel de symfony.

```
// Champ de saisie (input)
<?php echo input_tag('nom', 'valeur par défaut') ?>
// Tous les helpers supportent des paramètres
additionnels.
// Ils vous permettent de spécifier des attributs au
code généré
```

```
<?php echo input_tag('nom', 'valeur par défaut',
'maxlength=20') ?>
// Champ texte multiligne (text area)
<?php echo textarea_tag('nom', ' valeur par défaut',
'size=10x20') ?>
// Case à cocher (Check box)
<?php echo checkbox_tag('coche moi', 1, true) ?>
// Sélecteur (Radio button)
<?php echo radiobutton_tag('status[]', 'value1',
true) ?>
<?php echo radiobutton_tag('status[]', 'value2', false)
?>
// Liste déroulante (Combo)
<?php echo select_tag('paiement',
options_for_select(array('Visa', 'Eurocard',
'Mastercard')
)
?>
// Envoi de fichier
<?php echo input_file_tag('name') ?>
// Champ mot de passe
<?php echo input_password_tag('name', 'value') ?>
// Champ caché
<?php echo input_hidden_tag('name', 'value') ?>
// Bouton validation (texte)
<?php echo submit_tag('Save') ?>
// Bouton validation (image)
<?php echo submit_image_tag('submit_img') ?>
```

7.4.2. Les champs en liaison avec la base de données

Ces champs sont automatiquement remplis avec la valeur de la base de données. Vous devez pour cela préciser la variable qui contient l'objet enregistrement ainsi que l'accesseur :

```
<?php echo object_input_tag($client, 'getTelephone') ?>
```

Voici la liste des helpers objets :

```
<?php echo object_input_tag($object, $method, $options)
?>
<?php echo object_input_date_tag($object, $method,
$options) ?>
<?php echo object_input_hidden_tag($object, $method,
$options) ?>
<?php echo object_textarea_tag($object, $method,
$options) ?>
<?php echo object_checkbox_tag($object, $method,
$options) ?>
<?php echo object_select_tag($object, $method,
$options) ?>
<?php echo object_select_country_tag($object, $method,
$options) ?>
<?php echo object_select_language_tag($object, $method,
$options) ?>
```

Les helpers objets sont massivement utilisés dans la generation de modules CRUD et des modules d'administration.

8. La génération du backend d'administration

On a déjà vu que symfony était capable de générer des modules avec les principales fonctionnalités (CRUD). Vous avez également la possibilité de créer une interface administration permettant de modifier la base de données. Pour commencer, créez une nouvelle application au sein de votre projet :

```
symfony init-app adminapp
```

Il vous faudra ensuite créer, pour chaque classe Propel, le module d'administration correspondant (avec une syntaxe similaire à la génération d'un module CRUD).

```
symfony propel-init-admin adminapp utilisateur
Utilisateur
```

Cette syntaxe vous permettra de créer ces modules avec la configuration par défaut. Ils sont entièrement personnalisables. La documentation officielle y consacre une section entière (Lien22).

9. Conclusion

Vous êtes arrivés au bout de ce tutorial, félicitations ! Il ne vous reste plus qu'à essayer par vous-même:)

Retrouvez le tutoriel de Christopher Maneu n en ligne : Lien23

Premiers pas en ColdFusion

Cet article est adressé aux personnes n'ayant jamais développé en ColdFusion et désirant apprendre les rudiments de ce langage.

1. Présentation de ColdFusion

ColfFusion est un langage à base de balises (comme HTML ou XML) créé en 1995 et actuellement supporté par Adobe.
La force de ce langage et évidemment l'argument commercial de Adobe est la rapidité de développement des pages.
ColdFusion est un langage payant, la dernière version à ce jour est ColdFusion 8.
Pour une présentation plus complète je vous invite à consulter ce lien.

2. Le langage

2.1. Le célèbre "Hello world !"

Une fois n'est pas coutume notre premier exemple affichera simplement le texte : "Hello world!"

```
<cfoutput>
	Hello world !
</cfoutput>
```

La balise <cfoutput> permet d'écrire sur une page.

2.2. Déclaration et utilisation de variables

En coldfusion comme dans pas mal d'autres langages les variables ne sont pas typées.
La balise <cfset> permet de définir et d'attribuer une valeur à une variable.

```
<cfset i=1>
<cfset str="toto">
```

Affichons maintenant les valeurs de nos deux variables :

```
Ma première variable est un entier et sa valeur est : #i#<br>
Ma deuxième variable est une string et sa valeur est : #str#
```

En entourant la nom de ma variable par le symbole # je spécifie que je veux afficher le contenu de ma variable.
Si j'avais omis le # seul le nom de la variable aurait été affichée.
Vous aurez remarqué que la balise <cfoutput> accepte les balises HTML.

2.3. Les opérateurs de comparaison

- eq est égal
- neq est différent
- gt plus grand que
- lt plus petit que

2.4. Les opérateurs conditionnels

Les opérateurs conditionnels de ColfFusion sont <cfif>,<cfelse> et <cfelseif>

```
<cfif a gt b>
	<cfoutput>#a# est plus grand que #b#</cfoutput>
</cfif>
```

Nous allons maintenant ajouter un <cfelse>

```
<cfif a gt b>
	<cfoutput>#a# est plus grand que #b#</cfoutput>
<cfelse>
	<cfoutput>#a# est plus petit que #b#</cfoutput>
</cfif>
```

Et pour finir un exemple complet: <cfif> <cfelseif> <cfelse>

```
<cfif a gt 10>
	<cfoutput>#a# est plus grand que 10</cfoutput>
<cfelseif a gt 5>
	<cfoutput>#a# est plus grand que 5</cfoutput>
<cfelse>
	<cfoutput>#a# est plus petit que 5</cfoutput>
</cfif>
```

2.5. La boucle FOR

Une boucle peut être définie grâce à la balise <cfloop>

```
<cfloop index="i" from="1" to="10" step="1">
	<cfoutput>#i#<br></cfoutput>
</cfloop>
```

Cette boucle affiche les nombres de 1 à 10 l'un en dessous de l'autre.

3. Application.cfm et onrequestend.cfm

Ces deux noms de fichiers sont des noms réservés reconnu par le serveur ColdFusion.
Lors de l'exécution d'une page, le serveur commence par regarder si un fichier nommé Application.cfm existe dans le même répertoire, s'il le trouve il commence par exécuter le code qu'il contient.
De la même manière si le serveur trouve un fichier nommé onrequestend.cfm il l'exécute juste après la page demandée.

Dans cet exemple, je vais utiliser la page application.cfm afin de définir un header commun à toutes les pages

```
<cfapplication name="test">
<strong>HEADER FROM APPLICATION.CFM</strong><br>
```

La première ligne définit un nom pour que l'application soit

reconnue de manière unique auprès du serveur.
Elle est obligatoire si l'on travaille avec des variables de session.

De la même manière je vais créer un fichier nommé onrequestend.cfm qui définira un bas de page commun.

```
<br>
<strong>FOOTER FROM ONREQUESTEND.CFM</strong>
```

4. Gestion d'erreur

ColdFusion offre la possibilité d'intercepter et de gérer les erreurs pouvant survenir sur une page grâce aux balises <cftry> et <cfcatch>
Le code risquant de provoquer une exception doit être placé dans une balise <cftry>
Le code placé dans la balise <cfcatch> sera exécuté uniquement si une exception survient.

```
<cfset a = 2>
<cfset b = 0>
<cftry>
	<cfoutput>#a / b#</cfoutput>
<cfcatch>
	<cfoutput>Erreur : Division par zéro !
</cfoutput>
</cfcatch>
</cftry>
```

5. Utilisation de formulaire HTML

5.1. Exemple de GET

Dans cet exemple nous allons créer une page nommée "formGet.cfm" qui contient un form HTML pointant vers la page elle-même.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">

<html>
<head>
	<title>Untitled</title>
</head>

<body>
<cfif isdefined("url.maValeur")>
	<cfoutput>La valeur passée est :
#url.maValeur#</cfoutput>
<cfelse>
	<form name="formGet" action="formGet.cfm"
method="GET">
		<input type="Text" name="maValeur"><br>
		<input type="Submit" value="Ok">
	</form>
</cfif>
</body>
</html>
```

Examinons ce code :
Nous commençons par un **<cfif isdefined("url.maValeur")>** qui permet de tester si la variable url.maValeur est définie
url est un objet permettant d'accéder très facilement aux paramètres passés par l'url.
Ensuite si la variable n'est pas définie, on affiche un formulaire, dans le cas contraire on affiche la valeur passée par l'url.

5.2. Exemple de POST

Nous allons maintenant faire la même chose que dans le point 5.1 mais en utilisant un POST.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">

<html>
<head>
	<title>Untitled</title>
</head>

<body>
<cfif isdefined("form.maValeur")>
	<cfoutput>La valeur passée est :
#form.maValeur#</cfoutput>
<cfelse>
	<form name="formPost" action="formPost.cfm"
method="POST">
		<input type="Text" name="maValeur"><br>
		<input type="Submit" value="Ok">
	</form>
</cfif>
</body>
</html>
```

Le fichier est quasi identique. Seule l'action dans le form change et la manière de récupérer la valeur
Dans un post la valeur n'est pas passée par l'url on n'utilise donc pas l'objet url mais form

6. Envoyer un mail

Nous allons maintenant décrire comment faire pour envoyer un mail depuis le site en utilisant le client SMTP du serveur
Encore une fois ColdFusion nous a prémâché le travail en intégrant la balise <cfmail>

```
<cfmail from="toto@toto.com" to="titi@titi.com"
subject="Sujet du mail">
Contenu du mail
à envoyer
</cfmail>
```

Rien de plus simple ...

7. Travailler avec une base de données

Voici un point très important qui fait toute la puissance de ColdFusion : La facilité avec laquelle on peut travailler avec une base de données.
Avant de pouvoir travailler avec une base de données il faut commencer par créer une connexion vers la db dans votre IDE.
La manipulation étant différente suivant l'IDE utilisé, je ne peux que vous conseiller de vous référer à l'aide de votre application.

7.1. Lire les données d'une table

Nous allons dans cette exemple afficher le résultat d'une requête SELECT à l'aide de la balise <cfdump>
<cfdump> permet d'afficher n'importe quel type de variable (simple ou complexe) sur une page.
Parfait pour le debug.

```
<CFQUERY NAME="bookList" DATASOURCE="cfbookclub">
	SELECT * from BOOKS
</CFQUERY>

<cfdump var="#booklist#">
```

Voici comment macromedia justifie le prix de Coldfusion : vous affichez le contenu d'une base de données en quelques lignes de

code !

<cfdump> étant plutôt indiqué pour le debug nous allons faire la même chose au travers d'un tableau HTML :

```
<CFQUERY NAME="bookList" DATASOURCE="cfbookclub">
	SELECT * from BOOKS
</CFQUERY>

<table border="1">
<cfoutput query="booklist">
	<tr>
	<td>#booklist.BOOKID#</td>
	<td>#booklist.TITLE#</td>
	</tr>
</cfoutput>
</table>
```

Comment balayer toutes les lignes d'une query ? Avec <cfoutput>; tout simplement !
En effet la balise<cfoutput> accepte un argument query qui permet de spécifier quelles données parcourir.
Ensuite les champs récupérés de la db sont accessibles à partir d'un objet portant le même nom que votre query.

7.2. Exécuter une commande SQL

L'envoi d'une commande tel que INSERT, UPDATE, DELETE se fait exactement de la même manière que le SELECT.

```
<CFQUERY NAME="bookList" DATASOURCE="cfbookclub">
	UPDATE books SET author='toto' where id='4'
</CFQUERY>
```

Retrouvez l'article de Ludovic Lefort en ligne : Lien24

Comprendre et utiliser les interpolateurs

Dans ce tutoriel, nous allons comprendre et voir comment utiliser des interpolateurs sur des objets 3D créés via la librairie Actionscript 2 Sandy.

1. Introduction

Vous voilà déja peut-être plus intéressés par Sandy. Mais certainement vous vous dîtes, mais comment faire tourner un cube sur lui même en continu ? Réponse: les interpolateurs (interpolators).

2. Comprendre les interpolateurs

Les interpolateurs s'inscrivent comme des feuilles, et plus précisement comme des enfants de TransformGroup. Pourquoi ? Car un interpolateur va venir modifier la valeur de la transformation associée à son noeud parent en continu. Plus en détail, l'interpolateur va venir mettre à jour la matrice (tiens, vous voyez ce mot pour la première fois alors qu'on parle de 3D depuis 4 tutoriels maintenant !) de transformation au fur et à mesure du rendu.

Par conséquent, il est inutile d'associer une instance de Transform3D par setTransform à un TransformGroup qui possède un Interpolateur ! L'interpolateur va venir automatiquement écraser l'ancienne valeur.

Bon maintenant, voyons quels sont les interpolateurs dont Sandy dispose :

- **PositionInterpolator** : Comme son nom l'indique cet interpolateur permet de jouer sur la position de l'objet. Vous pourrez donc déplacer votre objet entre deux positions (deux instances de la classe Vector) sur un certain nombre de frames (nombre de rendus nécessaires à réaliser l'entière interpolation).
- **RotationInterpolator** : Permet de faire une rotation en continu entre deux angles sur un nombre de frames et selon un axe de rotation donné. Vous pouvez changer l'axe de rotation ainsi que le centre de rotation (point autour duquel l'objet va tourner) à tout moment.
- **ScaleInterpolator** : Permet d'appliquer des opérations de redimensionement sur l'objet. Etant donné que l'on peut déformer un objet 3D selon 3 axes, il faut renseigner les valeurs de scale pour chacune d'entre elles. Cela se fait par le biais de l'objet Vector. Ainsi pour définir le scale de départ et d'arrivée, il vaut passer en paramètre deux Vectors.

3. 1er exemple

3.1. Code

Réalisons maintenant notre première applications avec ces fameux interpolateurs. Nous utilisons ici un RotationInterpolator et nous allons faire faire à notre cube un tour sur lui-même selon l'axe vertical (les valeurs par defaut !)

```
import sandy.core.data.*;
import sandy.core.group.*;
import sandy.primitive.*;
import sandy.view.*;
import sandy.core.*;
import sandy.skin.*;
import sandy.util.*;
import sandy.core.transform.*;
import sandy.events.*;

function init( Void ):Void
{
	// nous créons notre ecran. Element primordial puisque c'est celui dans lequel les données seront affichées.
	var ecran:ClipScreen = new ClipScreen( this.createEmptyMovieClip('ecran', 1), 300, 300 );
	// nous créons notre camera. Le second paramètre est la distance nodale, qui donne donc l'effet
	// de perspective aux objets affichés
	var cam:Camera3D = new Camera3D( 700, ecran);
	// nous ajoutons notre camera au monde 3D
	World3D.getInstance().addCamera( cam );
	// nous créons notre Group qui servira de racine
	var bg:Group = new Group();
	// et l'ajoutons au monde 3D. Par défaut le dernier branchGroup ajouté est celui qui est actif
	// lors de la demande de rendu
	World3D.getInstance().setRootGroup( bg );
	// nous lançons la creation de la scene
	createScene( bg );
	// maintenant que tout est créé nous pouvons lancer le rencu du monde
	World3D.getInstance().render();
}

function createScene( bg:Group ):Void
```

```
{
        // nous créons notre objet. Pour cela nous
utilisons ici la primitive box.
        // cela créé une boite qui, dans le cas où les
dimensions d'hauteur, profondeur, largeur, sont les
memes
        // genère un cube. Ici nous créeons donc un cube
de 50 pixels.
        var o:Object3D = new Box( 50, 50, 50, 'quad' );
        // on créé un skin qui va permettre d'habiller
un peu mieux notre objet 3D.
        // ici le skin SimpleColor qui demande une
couleur de replissage, la valeur de l'alpha et
l'épaisseur du trait.
        var skin:Skin = new MixedSkin( 0x00FF00, 100, 0,
100 );
        // on applique le skin sur l'objet.
        o.setSkin( skin );
        // Nous créeons deux transform Group différents.
Chacun correspondant à un noeud de la branche de
transformations
        // que nous voulons creer.
        var tg1:TransformGroup = new TransformGroup();
        var tg2:TransformGroup = new TransformGroup();
        // Nous créeons les objets permettant de
transformer notre cube
        // Profitons-en d'ailleurs pour commencer à
nommer nos variables correctement. Vous le verrez ceci
        // deviens vite très important
        var translation:Transform3D = new Transform3D();
        // creation d'un interpolateur à la place du
transform3D
        var ease:Ease = new Ease();
        ease.linear();
        var rotint:RotationInterpolator = new
RotationInterpolator( ease.create(), 200 );
        // translation de 500 pixels dans l'axe des Z.
        translation.translate( 0, 0, 500 );
        // Nous appliquons la translation au noeud 1
        tg1.setTransform( translation );
        // Pour le noeud de rotation, nous n'appliquons
plus la transformation comme avant avec le
setTransform,
        // mais nous ajoutons un fils à ce noeud
        tg2.setTransform( rotint );
        // nous ajoutons notre objet comme enfant du
noeud correspondant à la rotation.
        // l'opération de rotation lui sera donc
appliquée avant celle de translation
        tg2.addChild( o );
        // Maintenant nous relions nos deux noeuds, en
mettant la translation en opération parent à celle de
rotation.
        // Cet ordre est très imporant selon le résultat
voulu!
        tg1.addChild( tg2 );
        // nous ajoutons le transformGroup comme filds
du BranchGroup afin de créer l'arborescence
        //et rendre l'object affichable.
        bg.addChild( tg1 );
}
// Nous lançons la création du monde 3D
init();
```

4. Savoir réagir aux événements

Nous venons de voir que nous sommes capables de faire tourner notre cube simplement. Mais bon, peut-être qu'il serait interessant de lui faire comprendre de continuer cette rotation continuellement ou sinon, qu'à la fin, notre interpolateur fasse marche arrière.

Et bien rassurez vous, cela est complètement faisable. Il suffit juste de comprendre comment réagir aux événements que vous envoient ces interpolateurs !

Voici une liste des événements qu'ils vous envoient :

- *onProgressEVENT* : envoyé durant la progression de l'interpolation
- *onPauseEVENT* : envoyé lorsque l'interpolateur est mis en pause
- *onResumeEVENT* : envoyé lorsque l'interpolateur sort du statut de pause
- *onStartEVENT*: envoyé lorsque l'interpolation commence
- *onEndEVENT* : envoyé lorsque l'interpolateur finit son interpolation.

Dans le prochain exemple, nous allons nous servir du onEndEVENT. Comment cibler cet événement ? Vous pouvez le faire de plusieurs façons :

- soit par la classe InterpolationEvent et du coup InterpolationEvent.onEndEVENT,
- soit par votre interpolateur, RotationInterpolator.onEndEVENT

5. La souscription

Vous pouvez souscrire à un événement aussi simplement que cela :

```
var rotint:RotationInterpolator = new
RotationInterpolator( tg2, ease.create(), 100 );
rotint.addEventListener( InterpolationEvent.onEndEVENT,
this, onRotationEnd );
```

Qu'est ce que c'est que onRotationEnd ? Si vous n'êtes pas familier avec les événements, peut-être vous posez vous cette question. Les autres, vous devez vous demander quoi mettre dans cette fonction.

Cette fonction est celle qui est appelée lorsque l'événement auquel vous vous êtes abonné est envoyé. Dans cette fonction est passé en paramètre un objet, qui dans le cas des interpolateurs est typé InterpolationEvent. Cet objet permet de retrouver l'objet émetteur, donc l'interpolateur, grâce à sa méthode getTarget().

Mais maintenant comment lui dire de recommencer ou d'aller en chemin inverse ? Et bien une fois que vous avez récupéré votre interpolateur, vous avez le choix entre deux méthodes :

redo() : Relance exactement la même interpolation
yoyo() : Lance l'interpolation en sens inverse

6. Deuxième exemple

6.1. Code

Je pense que vous avez maintenant les éléments en main pour comprendre le deuxième exemple :

```
import sandy.core.data.*;
import sandy.core.group.*;
import sandy.primitive.*;
import sandy.view.*;
import sandy.core.*;
import sandy.skin.*;
import sandy.util.*;
import sandy.core.transform.*;
import sandy.events.*;

function init( Void ):Void
```

```
{
	// nous créons notre ecran. Element primordial puisque c'est celui dans lequel les données seront affichées.
	var ecran:ClipScreen = new ClipScreen( this.createEmptyMovieClip('ecran', 1), 300, 300 );
	// nous créons notre camera. Le second paramètre est la distance nodale, qui donne donc l'effet
	// de perspective aux objets affichés
	var cam:Camera3D = new Camera3D( 700, ecran);
	// nous ajoutons notre camera au monde 3D
	World3D.getInstance().addCamera( cam );
	// nous créons notre Group qui servira de racine
	var bg:Group = new Group();
	// et l'ajoutons au monde 3D. Par défaut le dernier branchGroup ajouté est celui qui est actif
	// lors de la demande de rendu
	World3D.getInstance().setRootGroup( bg );
	// nous lançons la creation de la scene
	createScene( bg );
	// maintenant que tout est créé nous pouvons lancer le rencu du monde
	World3D.getInstance().render();
}

function createScene( bg:Group ):Void
{
	// nous créons notre objet. Pour cela nous utilisons ici la primitive box.
	// cela créé une boite qui, dans le cas où les dimensions d'hauteur, profondeur, largeur, sont les memes
	// genère un cube. Ici nous créeons donc un cube de 50 pixels.
	var o:Object3D = new Box( 50, 50, 50, 'quad' );
	// on créé un skin qui va permettre d'habiller un peu mieux notre objet 3D.
	// ici le skin SimpleColor qui demande une couleur de replissage, la valeur de l'alpha et l'épaisseur du trait.
	var skin:Skin = new MixedSkin( 0x00FF00, 100, 0, 100 );
	// on applique le skin sur l'objet.
	o.setSkin( skin );
	// Nous créeons deux transform Group différents. Chacun correspondant à un noeud de la branche de transformations
	// que nous voulons creer.
	var tg1:TransformGroup = new TransformGroup();
	var tg2:TransformGroup = new TransformGroup();
	// Nous créeons les objets permettant de transformer notre cube
	// Profitons-en d'ailleurs pour commencer à nommer nos variables correctement. Vous le verrez ceci
	// deviens vite très important
	var translation:Transform3D = new Transform3D();
	// creation d'un interpolateur à la place du transform3D
	var ease:Ease = new Ease();
	ease.linear();
	var rotint:RotationInterpolator = new RotationInterpolator( ease.create(), 100, 45, 180 );
	// On ajoute un écouteur à l'interpolateur pour se tenir au courant quand il a finit son trajet
	rotint.addEventListener( InterpolationEvent.onEndEVENT, this, onRotationEnd );
	// translation de 500 pixels dans l'axe des Z.
	translation.translate( 0, 0, 500 );
	// Nous appliquons la translation au noeud 1
	tg1.setTransform( translation );
	// Pour le noeud de rotation, nous n'appliquons plus la transformation comme avant avec le setTransform,
	// mais nous ajoutons un fils à ce noeud
	tg2.setTransform( rotint );
	// nous ajoutons notre objet comme enfant du noeud correspondant à la rotation.
	// l'opération de rotation lui sera donc appliquée avant celle de translation
	tg2.addChild( o );
	// Maintenant nous relions nos deux noeuds, en mettant la translation en opération parent à celle de rotation.
	// Cet ordre est très imporant selon le résultat voulu!
	tg1.addChild( tg2 );
	// nous ajoutons le transformGroup comme filds du BranchGroup afin de créer l'arborescence
	//et rendre l'object affichable.
	bg.addChild( tg1 );
}

// notre fonction qui reçoit l'evennement comme quoi la rotation est finie
function onRotationEnd( e:InterpolationEvent ):Void
{
	e.getTarget().yoyo();
}

// Nous lançons la création du monde 3D
init();
```

Retrouvez l'article de Thomas Pfeiffer avec le rendu des exmples en ligne : Lien24

Synthese vocale et .Net 3.0 : donnez la parole à vos applications sous Vista

1. Introduction

Il faut tout d'abord bien séparer synthèse et reconnaissance vocale. La synthèse est la génération d'une voix synthétique à partir d'un texte donné, alors que la reconnaissance consiste à reconnaitre et à retranscrire sous forme textuelle la parole d'un utilisateur. Bien que la solution de Microsoft sache faire les deux, nous allons ici nous intéresser à la synthèse, la reconnaissance fera l'objet d'un prochain article.

Le code sera proposé en VB.Net et en C#, et je vous le rappelle bien que basé sur le Framework .Net 3.0 ce qui sera abordé dans cet article ne sera garantit que pour un fonctionnement sous Vista, en effet la version de SAPI sur laquelle repose le code de cet article est la 5.3, qui est uniquement disponible sur le nouveau système d'exploitation de la firme de Redmond. D'ailleurs Vista utilise cette version 5.3 pour ses fonctionnalités de reconnaissance et de synthèse vocale.

2. Presentation de System.Speech

Une fois n'est pas coutume, en .Net tout se passe dans des namespaces, ici il se nomme System.Speech. Il contient tout ce dont nous aurons besoin pour synthétiser une voix et ainsi donner la parole à notre application. Vous devez donc tout d'abord ajouter une référence vers l'assembly le contenant, il s'agit de System.Speech.dll. Il ne vous reste plus qu'à ajouter Imports System.Speech.Synthetizer dans votre fichier de code pour ne pas avoir à taper à chaque fois les noms complets des objets que vous allez utiliser.

Ce namespace s'appuie sur la librairie COM SAPI 5.3 et en reprend la quasi totalité des fonctionnalités, mais cependant quelques unes ne sont pas présentes, je vous renvoie à la documentation officielle pour en avoir la liste. Si vous êtes adepte comme moi de .Net et pas spécialement de COM je vous recommande d'utiliser System.Speech.Synthetizer, les quelques fonctionnalités absentes par rapport a la libraire COM ne sont pas vitales.

Voici les classes majeures de ce namespace :

- **SpeechSynthetizer** : c'est la classe la plus importante. Elle contient toutes les methodes permettant de synthetiser la voix.
- **PromptBuilder** : permet de creer un document qui va servir a synthetiser la voix par l'intermediaire de SpeechSynthetizer. Cette classe permet aussi de serialiser ce document au format SSML.
- **PromptStyle** : cette classe permet de definir un style a appliquer a un document PromptBuilder, cela pour comprendre la vitesse, le volume, la pronomciation...

3. Donnons la parole a notre application

Commençons par l'exemple le plus simple qui soit, le célébrissime "Hello World". Nous allons tout d'abord ajouter une référence vers System.Speech.dll, puis ajouter Imports System.Speech.Synthetizer dans notre classe. A ce stade il nous faut désormais instancier un objet de type SpeechSynthetizer, c'est lui que nous allons utiliser pour synthétiser la voix, pour cela cette classe possède la méthode Speak() qui prend en paramètre le texte qui doit être synthétisé. Voici le code qui nous permet de faire parler notre application :

Hello World en VB.Net

```
Dim s As New SpeechSynthesizer
s.Speak("Hello World")
```

Hello World en C#

```
SpeechSynthesizer s = new SpeechSynthesizer();
s.Speak("Hello World");
```

Comme vous pouvez le voir c'est d'une complexité folle ! Placez ce code par exemple dans l'événement Click d'un bouton et il ne reste plus qu'à écouter la voix "mélodieuse" de votre ordinateur.

Mais voyons plus en détail ce qu'il est possible de faire grâce au Framework 3.0 et plus particulièrement au namespace System.Speech, vous allez voir c'est très sympathique et cela peut apporter un vrai plus à vos applications.

4. Allons plus loin

Jusqu'ici nous nous sommes contentés de synthétiser de la voix sans réel contrôle. Nous allons voir que nous pouvons par exemple enregistrer cette voix dans un fichier, gérer les paramètres de prononciation, le volume, la vitesse. Une des nouveautés et amélioration de SAPI en version 5.3 est le support du standard W3C SSML (Speech Synthesis Markup Language) qui permet d'établir facilement les paramètres (vitesse, prononciation, volume, pauses, intonations...) de la voix synthétisée. SSML s'appuie sur un ensemble de balises à la manière de HTML/XML. Ainsi, au lieu de passer une simple chaine de caractères nous passons une chaine au format SSML qui contient tous les paramètres souhaités. Un exemple de chaine au format SSML pourrait être le suivant :

```
<speak version="1.0"
xmlns="http://www.w3.org/2001/10/synthesis"
xml:lang="fr-FR">Hello world, welcome on the<sub
alias="dot net">.Net</sub>planet!</speak>
```

4.1. Choisir la voix à utiliser

Ici pas de longs discours non plus, car tout d'abord c'est très simple à réaliser et surtout le choix n'est pas des plus large, en effet pour l'instant il n'existe sous Vista qu'Anna pour l'anglais, Lili pour le chinois (à condition que votre Vista dispose du bon pack de langage), et Sam pour l'anglais sous XP, ce qui n'est pas dans le cadre de cet article. Cependant on peut espérer que Microsoft élargira ce choix, et de plus il toujours est possible d'ajouter si besoin de nouvelles voix à votre système d'exploitation, des éditeurs de logiciels spécialisés en proposent. Voici comment procéder depuis votre application .Net :

Choisir la voix à utiliser en VB.Net

```
Private Sub SelectVoice(ByVal name As String)
        ' La seule voix actuellement disponible sous
Vista Fr est "Microsoft Anna"
        Dim s As New SpeechSynthesizer
        s.SelectVoice(name)
        s.Speak("Hello, my name is Anna")
End Sub
```

Choisir la voix à utiliser en C#

```
private void SelectVoice(string name){
        // La seule voix actuellement disponible sous
Vista Fr est "Microsoft Anna"
        SpeechSynthesizer s = new SpeechSynthesizer();
        s.SelectVoice(name);
        s.Speak("Hello, my name is Anna");
}
```

4.2. Utilisation de la classe PromptBuilder

Si l'on veut éviter d'avoir à écrire cette chaine un peu barbare sans aucune assistance nous pouvons utiliser la classe PromptBuilder qui permet entre autre de générer une chaine SSML, mais aussi de définir les paramètres que l'on veut imposer à la voix synthétisée. En fait, PromptBuilder permet de lire, de générer du SSML, mais aussi de prendre en entrée des chaines de caractères simples, des chaines utilisant l'alphabet phonétique, etc. Il s'agit de la pierre angulaire de System.Speech lorsque l'on veut synthétiser de la voix de manière précise. Un exemple valant souvent mieux qu'un long discours voici ce qu'il est possible de faire en mettant en œuvre certaines possibilités de PromptBuilder.

Exemple d'utilisation de PromptBuilder en VB.Net

```
Private Sub PromptBuilderSample()
        Dim p As New PromptBuilder()

        Dim s1 As New PromptStyle
        s1.Volume = PromptVolume.ExtraLoud
        s1.Rate = PromptRate.Slow
        s1.Emphasis = PromptEmphasis.Reduced

        p.StartStyle(s1)

        p.StartParagraph()
        p.StartSentence()
        p.AppendText("Hello world, its's")
        p.AppendTextWithHint("07/10/2007", SayAs.Date)
        p.EndSentence()
        p.EndParagraph()

        p.EndStyle()

        p.AppendBreak(New TimeSpan(10000000))

        Dim s2 As New PromptStyle
        s2.Rate = PromptRate.Medium
        s2.Volume = PromptVolume.ExtraLoud
        s2.Emphasis = PromptEmphasis.Reduced

        p.StartStyle(s2)

        p.StartParagraph()
        p.AppendText("welcome on the")
        p.AppendTextWithAlias(".Net", "dot net")
        p.AppendText("planet!")
        p.EndParagraph()

        p.StartParagraph()
        p.StartSentence()
        p.AppendTextWithHint("VB", SayAs.SpellOut)
        p.AppendText("2005 is really a wonderful
language !")
        p.EndSentence()

        p.StartSentence()
        p.AppendTextWithHint("SSML", SayAs.SpellOut)
        p.AppendText("standard is now included in
SAPI")
        p.AppendTextWithHint("5.3", SayAs.SpellOut)
        p.EndSentence()
        p.EndParagraph()

        p.EndStyle()

        p.AppendBreak(New TimeSpan(10000000))
        p.AppendText("Please, visit my website at",
PromptRate.Slow)
        p.AppendTextWithHint("http://", SayAs.SpellOut)
        p.AppendText("webman dot developpez dot com")

        Dim synth As New SpeechSynthesizer
        synth.SetOutputToWaveFile("c:\voix.wav")
        synth.Speak(p)
        synth.SetOutputToNull()

End Sub
```

Exemple d'utilisation de PromptBuilder en C#

```
private void PromptBuilderSample(){
        PromptBuilder p = new PromptBuilder();

        PromptStyle s1 = new PromptStyle();
        s1.Volume = PromptVolume.ExtraLoud;
        s1.Rate = PromptRate.Slow;
        s1.Emphasis = PromptEmphasis.Reduced;

        p.StartStyle(s1);

        p.StartParagraph();
        p.StartSentence();
        p.AppendText("Hello world, its's");
        p.AppendTextWithHint("07/10/2007", SayAs.Date);
        p.EndSentence();
        p.EndParagraph();

        p.EndStyle();

        p.AppendBreak(new TimeSpan(10000000));

        PromptStyle s2 = new PromptStyle();
        s2.Rate = PromptRate.Medium;
        s2.Volume = PromptVolume.ExtraLoud;
        s2.Emphasis = PromptEmphasis.Reduced;

        p.StartStyle(s2);

        p.StartParagraph();
        p.AppendText("welcome on the");
        p.AppendTextWithAlias(".Net", "dot net");
        p.AppendText("planet!");
        p.EndParagraph();

        p.StartParagraph();
        p.StartSentence();
        p.AppendTextWithHint("VB", SayAs.SpellOut);
        p.AppendText("2005 is really a wonderful
language !");
        p.EndSentence();
```

```
        p.StartSentence();
        p.AppendTextWithHint("SSML", SayAs.SpellOut);
        p.AppendText("standard is now included in
SAPI");
        p.AppendTextWithHint("5.3", SayAs.SpellOut);
        p.EndSentence();
        p.EndParagraph();

        p.EndStyle();

        p.AppendBreak(new TimeSpan(10000000));

        p.AppendText("Please, visit my website at",
PromptRate.Slow);
        p.AppendTextWithHint("http://", SayAs.SpellOut);
        p.AppendText("webman dot developpez dot com");

        SpeechSynthesizer synth = new
SpeechSynthesizer();
        synth.SetOutputToWaveFile("c:\\voix.wav");
        synth.Speak(p);
        synth.SetOutputToNull();
}
```

Voici quelques membres de la classe PromptBuilder qui méritent que l'on s'attarde dessus. Cette liste n'est pas exhaustive, et je vous renvoie donc vers la documentation officielle dont le lien est disponible en fin d'article dans la partie "Ressources".

- **AppendAudio()** : cette méthode permet d'ajouter le contenu d'un fichier wav a notre objet PromptBuilder.
- **AppendText()** : cette méthode permet d'ajouter un texte simple à synthétiser, on peut passer en plus à la méthode la vitesse de diction souhaitée pour ce texte.
- **AppendTextWithHint()** : cette méthode permet également d'ajouter du texte, mais en plus elle permet de définir la manière de prononcer ce texte. Une énumération (SayAs) nous permet de choisir parmi une liste de prononciation définie, cela sera par exemple épeler le texte, le prononcer comme une date ou comme une heure…
- **AppendTextWithAlias()** : comme les deux précédentes cette méthode permet d'ajouter du texte a synthétiser mais elle permet de définir un alias qui correspondra a la manière de prononcer le texte qui peut être différente de la manière dont il est écrit. Un exemple simple pourrait être " .Net " que l'on veut prononcer comme " dot net ".
- **AppendBreak()** : cette méthode permet de marquer une pause, une rupture dans la lecture, qui peut être plus ou moins longue.
- **StartStyle() / End Style()** : permettent de définir un style (vitesse, volume, emphase…) pour le texte inclus entre ces deux méthodes. L'exemple que je donne un peu plus haut vous donne un bon aperçu de ce qu'il est possible de faire avec un objet PromptStyle.
- **StartParagraph()/EndParagraph()** : permettent de délimiter un paragraphe.
- **StartSentence()/EndSentence()** : permettent de délimiter une phrase.
- **SayAs** : cette énumération contient les prononciations prédéfinies que l'on peut entre autre spécifier en paramètre de la méthode AppendTextWithHint() que l'on vient de voir. Voici le contenu exhaustif de cette énumération : Date, Day, DayMonth, DayMonthYear, Month, MonthDay, MonthDayYear, MonthYear, NumberCardinal, NumberOrdinal, SpellOut, Telephone, Text, Time, Time12, Time24, Year, YearMonth, YearMonthDay

Je vous propose maintenant de voir comment PromptBuilder peut générer du SSML ou prendre du SSML en entrée. Tout d'abord, pour générer du SSML il faut utiliser la méthode ToXml() de la classe PromptBuilder, ca n'est pas plus complique que ca. Cette méthode retourne une chaine contenant le code SSML représentant la voix que l'on souhaite synthétiser. Si l'on veut conserver ces données SSML dans un fichier il suffit d'utiliser simplement un objet StreamWriter.

Exporter la chaine SSML directement dans un fichier en VB.Net

```
Private Sub PromptBuilderToSsmlFile()
        Dim p As New PromptBuilder()
        p.AppendText("Hello world, welcome on the")
        p.AppendTextWithAlias(".Net", "dot net")
        p.AppendText("planet!")

        Using sw As New
StreamWriter("c:\monfichier.ssml")
            sw.Write(p.ToXml())
        End Using
End Sub
```

Exporter la chaine SSML directement dans un fichier en C#

```
private void PromptBuilderToSsmlFile(){
        PromptBuilder p = new PromptBuilder();
        p.AppendText("Hello world, welcome on the");
        p.AppendTextWithAlias(".Net", "dot net");
        p.AppendText("planet!");

        using (StreamWriter sw = new
StreamWriter("c:\\monfichier.ssml")) {
                    sw.Write(p.ToXml());
        }
}
```

Remarque : attention, j'utilise ici Using qui s'occupe de la gestion du StreamWriter, dans d'autres circonstances il faudrait penser a fermer le StreamWriter avec la méthode Close().

Voyons comment maintenant importer du SSML directement dans notre objet PromptBuilder. Pour cela il existe deux méthodes qui sont AppendSsml() et AppendSsmlMarkup(). La première prends en paramètre le chemin d'un fichier contenant le code SSML, la deuxième quand a elle prend en directement en paramètre une chaine au format SSML. Voici un exemple d'utilisation de ces méthodes.

```
Private Sub AppendSsml()
        Dim p As New PromptBuilder()
        p.AppendSsml("c:\fichier.ssml")
        p.AppendSsmlMarkup("remplacer ce texte par
votre chaine SSML")

        Dim s As New SpeechSynthesizer
        s.Speak(p)

End Sub
```

```
private void AppendSsml(){
        PromptBuilder p = new PromptBuilder();
        p.AppendSsml("c:\\fichier.ssml");
        p.AppendSsmlMarkup("remplacer ce texte par votre
chaine SSML");

        SpeechSynthesizer s = new SpeechSynthesizer();
        s.Speak(p);
}
```

Remarque : dans cette partie nous venons de voir que l'objet

PromptBuilder est tres souple en termes de sources de texte à synthétiser, cela peut être des chaines des caractères simples, du SSML, du wav, en flux, en fichier, etc. Et tout cela est possible pour un même objet, il est donc envisageable de combiner différentes sources sans pour autant avoir à les contraindre préalablement dans un format donné. A l'usage cela se révèle très appréciable.

4.3. Persistance d'une voix synthetisee

Si l'on veut faire persister une voix synthétisée ou pourquoi pas l'échanger entre différentes machines et pourquoi pas plateforme, nous avons deux possibilités principales : enregistrer la voix dans un fichier Wav (le format étant supporté sur de nombreuses plateformes) ou la sérialiser au format SSML. La deuxième solution est évidemment dans la majorité des situations la meilleure, d'autant plus que SSML est un standard W3C largement adopté.

Enregistrement de la voix dans un fichier Wav :

Synthétiser un PromptBuilder dans un fichier Wav en VB.Net

```
Private Sub PromptBuilderToWavFile()
        Dim p As New PromptBuilder
        p.AppendTextWithHint("VB", SayAs.SpellOut)
        p.AppendTextWithAlias(".Net", "dot net")
        p.AppendText("is the nicest language for .Net
platform !")

        Dim synth As New SpeechSynthesizer
        synth.SetOutputToWaveFile("c:\voix.wav")
        synth.Speak(p)
        synth.SetOutputToNull()
End Sub
```

Synthétiser un PromptBuilder dans un fichier Wav en C#

```
private void PromptBuilderToWavFile(){
       PromptBuilder p = new PromptBuilder();
       p.AppendTextWithHint("VB", SayAs.SpellOut);
       p.AppendTextWithAlias(".Net", "dot net");
       p.AppendText("is the nicest language for .Net
platform !");

       SpeechSynthesizer synth = new
SpeechSynthesizer();
       synth.SetOutputToWaveFile("c:\\voix.wav");
       synth.Speak(p);
       synth.SetOutputToNull();
}
```

Sérialisation SSML de la voix :

Sérialisation d'un PromptBuilder en VB.Net

```
Private Function PromptBuilderToSsml() As String
        Dim p As New PromptBuilder
        p.AppendTextWithHint("VB", SayAs.SpellOut)
        p.AppendTextWithAlias(".Net", "dot net")
        p.AppendText("is the nicest language for .Net
platform !")

        Return p.ToXml
End Function
```

Sérialisation d'un PromptBuilder en C#

```
private string PromptBuilderToSsml(){
        PromptBuilder p = new PromptBuilder();
        p.AppendTextWithHint("VB", SayAs.SpellOut);
        p.AppendTextWithAlias(".Net", "dot net");
        p.AppendText("is the nicest language for .Net
platform !");

        return p.ToXml;
}
```

Remarque : attention au wav, il s'agit d'un format non compresse, et donc le volume du fichier peut rapidement augmenter, même s'il est possible d'ajuster les paramètres d'enregistrement il faudra garder à l'esprit ce détail. Voici un argument de plus en faveur d'une sérialisation SSML de la voix, plutôt que d'un enregistrement dans un fichier wav.

5. Conclusion

J'espère que cet article vous aura donné l'envie de tester la synthèse vocale sous Vista, et pourquoi pas même de l'intégrer dans vos prochaines applications. Il serait cependant souhaitable que Microsoft ajoute de nouvelles voix en plus de celles de Lili et d'Anna. Nous avons vu que System.Speech comporte une partie synthèse, mais aussi une partie reconnaissance vocale qui fera l'objet d'un prochain article.

Retrouvez l'article de Ronald Vasseur en ligne : Lien26

Surcharge, Redéfinition et Occultation avec VB.NET

L'objectif de ce tutorial est d'expliquer les notions de surcharge, de redéfinition et d'occultation apparues avec Visual Basic .NET, afin d'éviter aux développeurs toute confusion entre les mots clés servant à mettre en oeuvre ces mécanismes.

1. Introduction

.NET est la première version de Visual Basic à introduire la notion d'héritage entre classes. Cela a conduit à l'apparition de nouveaux mots clés dont certains se ressemblent par leur syntaxe et d'autres par leur fonctionnement. De ce fait, les développeurs peuvent se trouver facilement perdus.
Ainsi, si le concept de surcharge et le rôle du mot clé Overloads sont faciles à comprendre, la confusion entre Overloads et deux mots clés introduits par la redéfinition, Overridable et Overrides, est probable. Même pour les développeurs qui arrivent à passer ce stade sans incident, l'apprentissage de l'occultation et de son mot clé Shadows risque de tout chambouler ! Pour présenter les choses de manière claire, ce tutorial commence par expliquer les notions d'héritage et de polymorphisme nécessaires pour comprendre le reste du document. Ensuite, il aborde les mécanismes de surcharge, de redéfinition et d'occultation en mettant l'accent plus sur le concept que sur les mots clés. Les explications seront appuyées par des exemples simples et chaque paragraphe sera clôturé par une liste des points essentiels à retenir. Beaucoup de livres et articles traitent des concepts abordés dans ce document, le présent tutorial a pour but de faciliter l'assimilation et d'éviter les confusions habituelles. Pour cela, nous allons volontairement éviter certains détails, par exemple, certaines règles spécifiques aux constructeurs, aux évènements et aux membres partagés ne seront pas évoquées.

2. Définitions

Avec sa version .NET, Visual Basic est devenu un véritable langage orienté objet, puisque cette version a introduit les constructeurs de classes, l'héritage entre classes, le

polymorphisme et bien d'autres concepts. Même si ce tutorial suppose que vous connaissez les bases de l'orienté objet et la syntaxe correspondante dans Visual Basic .NET, nous allons revenir ici sur quelques définitions qui nous seront utiles par la suite.

2.1. L'héritage

L'héritage entre classes consiste en la possibilité de créer une classe, dite classe dérivée, qui englobe toutes les propriétés, les méthodes et les évènements définis dans une autre classe, dite classe de base, sans avoir à les re-saisir. Bien entendu, le principal avantage de l'héritage n'est pas d'éviter la saisie mais surtout d'établir un lien de spécialisation entre les classes. Par exemple, on peut créer une classe Employe en la faisant hériter d'une classe Personne, puisque tout employé est une personne. Ainsi, la classe Employe contiendra automatiquement toutes les caractéristiques de la classe Personne comme le nom et le prénom, et pourra définir des membres spécifiques à un employé comme le salaire par exemple.

Dans Visual Basic .NET, la mise en oeuvre de l'héritage passe par l'utilisation du mot clé Inherits, comme le montre l'exemple suivant : Déclaration de la classe de base :

```
Public Class Personne
Public titre As String
Public nom As String
Public prenom As String
Public DateDeNaissance As Date
End Class
```

Pour la classe dérivée on utilise le mot clé Inherits et on définit uniquement les membres spécifiques à un employé :

```
Public Class Employe
Inherits Personne
Public Salaire As Double
End Class
```

Comme la classe Employe englobe automatiquement tous les membres de la classe Personne, le code suivant fonctionne sans problème :

```
Module Execution
Sub main()
Dim e As New Employe
'On peut faire référence aux membres
'définis dans la classe Personne
Console.Out.WriteLine(e.nom & " " & e.prenom)
End Sub
End Module
```

Remarque : vous êtes supposés savoir qu'il est déconseillé d'utiliser des champs publics dans les classes, et qu'il faut utiliser des champs Private exposés, si nécessaire, via des propriétés. Dans ce tutorial, cette règle ne sera pas respectée dans le but de condenser le code.

2.2. Le polymorphisme

Le polymorphisme est la possibilité de référencer un objet du type de la classe dérivée avec une variable dont le type est la classe de base, comme dans l'exemple suivant où, à la variable p de type Personne, on affecte un objet Employe, ce code est logiquement correct puisque un employé est en fait une personne :

```
Dim p As Personne
p = New Employe
```

Le polymorphisme garantie aussi autre chose, si une méthode définie dans la classe de base est redéfinie dans une classe dérivée, l'appel à cette méthode à travers une variable de type classe de base va dépendre du type de l'objet pointé par la variable et non pas du type de la variable. Pour clarifier ces propos, considérons l'exemple suivant :

La classe Personne définit une méthode affiche() qui affiche les renseignements de la personne, la classe Employe qui dérive de Personne redéfinit la méthode affiche() afin d'afficher aussi le salaire de l'employé. Dans ce cas quel sera le résultat de cette portion de code

```
Dim p As Personne
p = New Employe
'Cette instruction affiche aussi le salaire car la
'méthode appelé est celle de la classe employe
p.affiche()
```

Dans la dernière instruction, est-ce que c'est la méthode définie dans la classe Personne qui sera appelée ou bien celle définie dans la classe Employe ?
Autrement dit, est-ce que le salaire sera affiché ou non ?
Le polymorphisme garantit aussi que, si une méthode appelée soit celle définie dans la classe de l'objet désigné par la variable et non pas celle de la classe de la variable utilisée pour appeler la méthode. Dans notre cas, la variable p utilisée pour appeler la méthode est de type Personne, mais comme elle pointe vers un objet Employe (à cause de l'instruction : p = new Employe) alors la méthode appelée est celle définie dans la classe Employe, et par conséquent le salaire sera affiché.

2.3. Signature de paramètres d'une méthode

La signature de paramètres d'une méthode est définie par le nombre, l'ordre et le type de ses paramètres. Par exemple, les deux méthodes suivantes ont la même signature :

```
Sub methode1(ByVal par1 As Integer, ByVal par2 As
String, ByRef par3 As String)
...
End Sub
Sub methode1(ByVal p1 As Integer, ByVal p2 As String,
ByVal p3 As String)
...
End Sub
```

Remarquez que le nom des paramètres et leur ordre de passage (byval ou byref) n'ont pas d'influence sur la signature. Pour que deux signatures de paramètres soit différentes, il ne faut pas que la différence soit due uniquement à un paramètre optionnel ou uniquement aux types de retour pour les fonctions. Dans les deux exemples suivants le compilateur va générer une erreur. Rappelons qu'il n'est pas possible de déclarer dans la même classe deux méthodes avec le même nom et la même signature de paramètres :

```
'Erreur les deux signatures diffèrent par un paramètre
optionnel, ce qui n'est pas suffisant
Sub methode1(ByVal param1 As Integer)
...
End Sub
Sub methode1(ByVal p1 As Integer, Optional ByVal p3 As
String = "")
...
End Sub
```

```
'Erreur les deux signatures diffèrent uniquement par
leur type de retour
'ce qui n'est pas suffisant
Function fonction1(ByVal param1 As Integer) As Integer
```

```
'...
End Function
Function fonction1(ByVal p1 As Integer) As Date
'...
End Function
```

Maintenant que vous avez appris ce que sont l'héritage, le polymorphisme et la signature de méthode, nous pouvons aborder la surcharge, la redéfinition et l'occultation sans risquer aucune ambiguïté.

3. La surcharge de méthodes

Visual Basic .NET autorise la surcharge de méthodes, c'est-à-dire qu'il permet de déclarer dans une même classe plusieurs méthodes ayant le même nom mais des signatures différentes.
Lorsque vous écrivez plusieurs versions d'une même méthode dans une classe, s'il n y a aucune méthode portant le même nom dans les classes parentes de cette classe 2 alors vous n'avez besoin d'aucun mot clé particulier pour déclarer vos méthodes, comme dans le cas suivant 3 pour la fonction additionne :

```
Public Class Addition
'Plusieurs version de la fonction additionne. aucun mot
'clé particulier n'est nécessaire 'puisque la classe
'Object ne possède pas une méthode additionne
Function additionne(ByVal p1 As Double, ByVal p2 As
Double) As Double
Return p1 + p2
End Function
Function additionne(ByVal p1 As Double, ByVal p2 As
Double, _
ByVal p3 As Double) As Double
Return p1 + p2 + p3
End Function
End Class
```

En revanche, si la méthode que l'on souhaite surcharger a déjà été définie dans l'une des classes de base, alors chaque version surchargée de la méthode dans la classe en cours doit être précédée par le mot clé Overloads.

Dans l'exemple suivant, nous allons écrire la classe AdditionAvancee qui hérite de la classe Addition précédente. Dans cette nouvelle classe, nous allons définir deux nouvelles version de la méthode Additionne, qui prennent respectivement quatre et cinq paramètres. Comme la méthode Additionne a déjà été définie dans la classe de base Addition, alors les deux nouvelles versions de la méthode Additionne doivent être précédées par le mot clé Overloads :

```
Public Class AdditionAvancee
Inherits Addition
Overloads Function additionne(ByVal p1 As Double, ByVal
p2 As Double, _
ByVal p3 As Double, ByVal p4 As Double) As Double
Return p1 + p2 + p3 + p4
End Function
Overloads Function additionne(ByVal p1 As Double, ByVal
p2 As Double, _
ByVal p3 As Double, ByVal p4 As Double, ByVal p5 As
Double) As Double
Return p1 + p2 + p3 + p4 + p5
End Function
End Class
```

Dans la classe fille, nous pouvons définir une méthode qui a le même nom et la même signature qu'une méthode de la classe de base en utilisant le mot clé Overloads. Cependant, comme elles ont la même signature ce ne sera pas une surcharge mais une occultation, nous reviendrons un peu plus loin sur ce concept.

A retenir

- Nous pouvons définir plusieurs versions d'une même méthode à condition qu'elles aient des signatures différentes. Dans la terminologie orientée objet cela s'appelle la surcharge de méthodes.
- Lorsque nous définissons une méthode surchargée pour la première fois, aucun mot clé n'est nécessaire. Les méthodes seront écrites normalement.
- Lorsque nous définissons des versions de surcharge pour une méthode qui a été définie dans l'une des classes de base alors, chaque version doit être précédée par le mot clé Overloads.
- La surcharge peut s'appliquer aussi aux propriétés, dans ce cas les règles sont les mêmes que pour les méthodes.

4. La redéfinition de méthodes

Comme vous le savez déjà, une classe dérivée hérite de toutes les méthodes et propriétés de sa classe de base. Or, il se pourrait qu'une méthode nécessite d'être adaptée à la classe dérivée et non pas reprise en l'état.

Considérons à nouveau les deux classes Personne et Employe :

```
Public Class Personne
Public titre As String
Public nom As String
Public prenom As String
Public Sub affiche()
Console.Out.Write(titre & " " & nom & " " & prenom)
End Sub
End Class
```

Voici le code de la classe Employe :

```
Public Class Employe
Inherits Personne
Public Salaire As Double
End Class
```

Dans le cas précédent, la classe Employe va hériter de la méthode affiche définie dans la classe Personne. Or cette méthode n'affiche pas le salaire de l'employé et donc elle est en quelque sorte incomplète pour la classe Employe qui a tout intérêt de l'adapter à ses besoins en la redéfinissant. Nous allons voir comment se fait la redéfinition de méthode.
Tout d'abord, il faut savoir que c'est le concepteur de la classe de base qui décide quelles méthodes peuvent être redéfinies. Par défaut, les méthodes ne peuvent pas être redéfinies. Elles doivent être uniquement héritées en l'état ou occultées (le chapitre suivant est consacrée à l'occultation). Pour déclarer une méthode qui peut être redéfinie il faut faire précéder sa déclaration par le mot clé Overridable. Par exemple, voici la déclaration de la méthode affiche() de la classe Personne pour qu'elle soit redéfinissable 4 :

```
Public Overridable Sub affiche()
Console.Out.Write(titre & " " & nom & " " & prenom)
End Sub
```

Maintenant que nous avons rendu la méthode affiche() virtuelle, nous allons voir comme la redéfinir dans la classe dérivée. Pour cela, il suffit de déclarer dans la classe dérivée une méthode qui a le même nom et la même signature 5 que la méthode virtuelle en faisant précéder sa déclaration par le mot clé Overrides. Comme dans l'exemple suivant pour la classe Employe :

```
Public Class Employe
Inherits Personne
```

```
Public salaire As Double
'La classe Personne possède maintenant sa propre
'méthode affiche qui affiche aussi le salaire
Public Overrides Sub affiche()
Console.Out.Write(titre & " " & nom & " " & prenom & "
" & salaire)
End Sub
End Class
```

Ce qui est important à savoir c'est que la redéfinition de méthodes est totalement compatible avec le polymorphisme. Ainsi, si on appelle la méthode affiche() à partir d'une variable de type Personne, ce n'est pas la méthode définie dans la classe Personne qui sera toujours appelée, car le comportement polymorphique veut que la méthode appelée ne soit pas celle du type de la variable mais du type pointé par la variable. Encore un exemple pour illustrer ces propos :

```
Module Execution
Sub main()
Dim p As New Personne
p.titre = "M."
p.nom = "Dubois"
p.prenom = "Dupont"
p.affiche()'Appelle la méthode affiche de Personne
Console.Out.WriteLine()
Dim e = New Employe
e.titre = "Mme"
e.nom = "Dupuis"
e.prenom = "Doloris"
e.salaire = 10000
p = e 'La variable p pointe désormais vers un objet
Employe
p.affiche()
End Sub
End Module
```

Dans cet exemple, la première instruction p.affiche() va faire appel à la méthode définie dans la classe Personne, non pas parce que la variable p est de type Personne mais parce qu'elle pointe vers un objet de type Personne. La deuxième instruction p.affiche() va faire appel à la méthode définie dans la classe Employe qui affiche aussi le salaire car la variable p pointe vers un objet Employe.

Lors de la redéfinition d'une méthode, l'utilisation du mot clé MyBase pour appeler la méthode de la classe de base peut souvent nous rendre un grand service. Dans l'exemple précédent la méthode affiche() de la classe Employe aurait pu être écrite comme suit :

```
Public Overrides Sub affiche()
MyBase.affiche()
Console.Out.Write(" " & salaire)
End Sub
```

Autre chose importante à signaler, les méthodes déclarées Overrides (c'est-à-dire qui redéfinissent une méthode de leur classe de base) sont automatiquement virtuelles sans qu'il y ait besoin d'utiliser le mot clé Overridable. Par exemple, une classe Manager qui hérite de la classe Employe peut redéfinir la méthode affiche même si cette méthode n'a pas été déclarée Overridable dans la classe Employe. Pour faire en sorte qu'une méthode redéfinie ne soit plus virtuelle il faut utiliser explicitement le mot clé NotOverridable. Voici le code complet et commenté de la classe Manager :

```
Public Class Manager
Inherits Employe
Public PrimeResponsabilite As Double
'La classe Manager peut redéfinir la méthode affiche de
'la classe Employe mais en la déclarant NotOverridable
'les classes qui hériteront de la classe Manager ne
'pourront plus redéfinir cette méthode !
Public NotOverridable Overrides Sub affiche()
MyBase.affiche()
Console.Out.Write(" " & PrimeResponsabilite)
End Sub
End Class
```

A retenir

- Pour rendre une méthode redéfinissable il faut faire précéder sa déclaration par le mot clés
- Overridable. Cette méthode sera appelée alors méthode virtuelle.
- Pour redéfinir une méthode virtuelle, il suffit de définir une méthode dans la classe dérivée qui a le même nom et la même signature que la méthode virtuelle et dont la déclaration est précédée par le mot clé Overrides.
- Les méthodes qui redéfinissent des méthodes virtuelles sont automatiquement virtuelles à leur tour. Pour faire en sorte qu'une telle méthode ne soit pas virtuelle il faut précéder sa déclaration par le mot clé NotOverridable.
- L'appel à une méthode redéfinie est complètement polymorphe.
- La redéfinition peut s'appliquer aussi aux propriétés, dans ce cas les règles sont les mêmes que pour les méthodes.

Et si on redéfinit une méthode surchargée ?

A ce stade, et avant de poursuivre il est important de comprendre exactement la différence entre surcharge et redéfinition et le rôle de chacun des mots clés Overloads, Overridable et Overrides.
Dans une classe dérivée, on peut à la fois redéfinir une méthode virtuelle (avec une méthode qui a le même nom et la même signature) et surcharger cette méthode (en proposant d'autres méthodes ayant le même nom mais des signatures différentes).
Ce cas de figure est en fait très simple, il n y a absolument pas de règles spécifiques, ce sont les règles précédentes qui s'appliquent. Ainsi :

- Dans la classe dérivée, toutes les versions surchargées de la méthode doivent être précédées par le mot clé Overloads.
- La méthode ayant la même signature que la méthode virtuelle doit être précédée par le mot clé Overrides (puisqu'il s'agit d'une redéfinition). Par conséquent, cette dernière méthode sera précédée à la fois par le mot clé Overloads et le mot clé Overrides.

En voici un exemple de la classe Employe qui redéfinie la méthode affiche et lui propose une nouvelle version surchargée :

```
Public Class Employe
Inherits Personne
Public salaire As Double
'Méthode de redéfinition et de surcharge
'Précédée à la fois par Overloads et Overrides
Public Overloads Overrides Sub affiche()
Console.Out.Write(titre & " " & nom & " " & prenom & "
" & salaire)
End Sub
'Méthode de surcharge seulement précédée par Overloads
Public Overloads Sub affiche(ByVal prefixe As String)
Console.Out.Write(prefixe & " ")
MyClass.affiche()
End Sub
End Class
```

Retrouvez la suite de l'article de Mohamed Elhadi Benzeghiba : Lien27

Les derniers tutoriels et articles

Gérer ses ressources de manière robuste en C++

En C++, il n'y a pas de destruction automatique des objets lorsque l'on perd leur trace dans le code source. Les objets ainsi perdus le sont définitivement, on parle alors de fuite. C'est donc au programmeur C++ qu'incombe l'entière responsabilité de gérer le cycle de vie des objets alloués. Il s'agit donc là d'une problématique centrale dans ce langage, qui doit être réfléchie et résolue de manière globale. C'est ce que des experts ont fait, et des techniques spécifiques apportant une réponse globale au problème de la gestion des ressources (et non pas seulement au cas particulier de la mémoire) ont été développées. Ces pratiques sont à la fois robustes et élégantes, mais restent cependant peu connues et sous-utilisées. Le but de cet article est d'accroître leur notoriété au travers de leur mise en oeuvre dans le cas d'un problème classique de gestion de ressource limitée.

1. Introduction

Prenons le cas d'une ressource quelconque manipulée au moyen de la classe Resource, dont le but est de représenter cette ressource d'un point de vue du langage et de permettre sa manipulation en C++. Il pourrait s'agir d'une connexion réseau par exemple. Notre classe de ressource s'appellerait alors certainement Socket et non pas Resource.

Maintenant, imaginez que vous deviez contrôler le nombre d'instances de cette classe Resource. Dans notre exemple avec la classe Socket, il pourrait s'agir de limiter le nombre de connexions réseaux simultanément ouvertes. Vous vous retrouvez alors avec la contrainte de devoir comptabiliser le nombre d'instances de Resource qui sont toujours en cours d'utilisation. A priori, cela parait enfantin. Mais comme toujours, et encore plus en C++, ce n'est pas parce qu'il y a de nombreuses façons de faire qu'elles se valent toutes. Penchons-nous donc vers un début de solution.

2. Approche classique

Dans le cadre de cet article, j'ai choisi de centraliser les opérations de création et de contrôle du nombre d'instances de Resource au sein d'un gestionnaire spécialisé baptisé ResourceManager.

Cette classe ResourceManager s'apparente en fait à une factory, et possède une fonction membre New() chargée d'instancier ou non une nouvelle ressource, et d'en conserver la trace si jamais c'est le cas. Le but est de savoir à tout moment le nombre d'instances en cours d'utilisation, afin de limiter leur nombre. Ce nombre maximal d'instances est fourni comme paramètre au constructeur de ResourceManager.

ResourceManager.h

```
class Resource;
typedef Resource* ResourcePtr;

class ResourceManager{
public:
    // Définit le nombre maximal de Resource
utilisables simultanément
    ResourceManager( int MaxNbInstances );

    // Renvoie une nouvelle Resource s'il y en a moins
que
    // MaxNbInstances en cours d'utilisation, NULL
sinon.
    ResourcePtr New();
};
```

Si concevoir cette interface publique d'utilisation de ResourceManager est une chose, l'implémenter en est une autre. Car un problème se pose : comment connaître le nombre d'instances créées (facile) et toujours en cours d'utilisation (un peu moins facile) ? Autrement dit, comment être informé dans ResourceManager qu'un objet alloué via New() a été détruit ?

Une première solution, fréquente en programmation procédurale, consiste à fournir une fonction paire de New, chargée de la destruction des instances :

ResourceManager.h

```
class Resource;
typedef Resource* ResourcePtr;

class ResourceManager{
public:
    // Définit le nombre maximal de Resource
utilisables simultanément
    ResourceManager( int MaxNbInstances );

    // Renvoie une nouvelle Resource s'il y en a moins
que
    // MaxNbInstances en cours d'utilisation, NULL
sinon.
    ResourcePtr New();

    // Supprime une Resource allouée avec New()
    void Delete( ResourcePtr );

private:
    // Nombre d'instances en cours d'utilisation
    int NbInstances;
    // Nombre maximal d'instances autorisées
    const int MaxNbInstances;
};
```

ResourceManager.cpp

```
#include "ResourceManager.h"
#include "Resource.h"

ResourceManager::ResourceManager( int Max ) :
```

```
    NbInstances( 0 ),
    MaxNbInstances( Max )
    {}

ResourcePtr ResourceManager::New(){
    if ( NbInstances < MaxNbInstances ){
        ResourcePtr r = new Resource();
        // On incrémente NbInstances _après_ avoir
alloué
        // la ressource avec succès, afin de laisser
notre
        // Manager dans un état cohérent si une
exception
        // est levée lors de la création d'un objet
Resource
        ++this->NbInstances;
        return r;
    }
    return 0; // NULL
}

void ResourceManager::Delete( ResourcePtr Res ){
    if ( Res ){
        --this->NbInstances;
        delete Res;
    }
}
```

Tout comme ResourceManager::New() est chargé d'instancier des objets de type Resource, ResourceManager::Delete() est chargé de les détruire. On pourrait envisager de confier davantage de travail à ces fonctions, comme par exemple maintenir une liste des instances allouées, mais j'ai préféré garder les choses aussi simples que possibles.

Notez l'ordre des opérations au sein de ResourceManager::New() : l'objet Resource est d'abord alloué, puis le compteur NbInstances est mis à jour, et surtout pas l'inverse. Car si l'allocation d'un nouvel objet Resource venait à échouer (par manque de mémoire par exemple) et qu'une exception était levée (std::bad_alloc), l'objet ResourceManager se retrouverait alors dans un état incohérent : un objet supplémentaire Resource serait comptabilisé comme en cours d'utilisation alors qu'il n'existe pas.

3. Le problème de cette approche classique

Ce concept de fonctions jumelles repose sur un principe de programmation très saint : C'est celui qui a alloué un objet qui devrait le détruire. ResourceManager est ainsi responsable de la création mais aussi de la destruction des objets Resource au moyen de son couple de fonctions membres New() et Delete().

Si ce principe est très saint, sa mise en oeuvre dans cette première tentative l'est un peu moins, et n'est pas vraiment ce qui se fait de mieux en C++. En effet, en introduisant la fonction Delete() dans l'interface publique de ResourceManager, nous avons ajouté une contrainte de taille quant à l'utilisation de la fonction New() : le pointeur qu'elle retourne doit obligatoirement et systématiquement être libéré au moyen de Delete().

Si dans certains langages, comme en C, ce n'est déjà pas une contrainte évidente à satisfaire en n'importe quelle circonstance (un oubli est si vite arrivé), cela devient particulièrement complexe en C++ à cause du support des exceptions par le langage.

```
// Fonction quelconque qui utilise un objet Resource
void UseResource( ResourcePtr );

void Test(){
    // on autorise 3 instances maximum
    ResourceManager mgr( 3 );

    ResourcePtr r = mgr.New();
    if ( r ) {
        UseResource( r );
        mgr.Delete( r );
    }
}
```

Dans l'exemple ci-dessus, si une exception est levée par la fonction UseResource(), le flot d'exécution classique est interrompu et le pointeur r renvoyé par ResourceManager::New() n'est donc jamais libéré. Autrement dit, la ressource est définitivement perdue!

De nombreux langages objets (Java, C#, Python, PHP, ...) remédient à ce problème au moyen du mot-clé **finally**, mais C++ n'en fait pas partie. Mais il est possible d'approcher sa philosophie d'utilisation au moyen de la construction suivante par exemple :

```
// Fonction quelconque qui utilise un objet Resource
void UseResource( ResourcePtr );

void Test(){
    // on autorise 3 instances maximum
    ResourceManager mgr( 3 );

    ResourcePtr r = mgr.New();
    if ( r ){
        try{
            UseResource( r );
        } catch ( ... ) { // pseudo bloc finally
            mgr.Delete( r );
            throw; // relancer l'exception
        }
        mgr.Delete( r );
    }
}
```

Comme on peut le voir ci-dessus, gérer les ressources de cette manière est extrêmement lourd. De plus, comme le C++ ne collecte pas automatiquement la mémoire au contraire des langages qui proposent le mot-clé finally, une telle construction serait à utiliser bien plus souvent que dans ces autres langages, ce qui rendrait le code tout simplement illisible, et inciterait le programmeur à négliger ce genre de "détails". L'utilisation de la construction **try...finally** n'est donc pas une réponse adaptée à un langage comme C++.

Qui plus est, confier ainsi autant de travail au programmeur client pour s'assurer que notre gestion d'objets soit valide est le signe d'une bien piètre robustesse de la part de notre classe ResourceManager. En effet : une utilisation trop basique de cette classe suffit à mettre en péril sa fiabilité, et **c'est au programmeur client de s'assurer de maintenir la stabilité de notre système!** Or, le C++ a la réputation de permettre l'écriture de programmes robustes. Oui, mais comment ?

4. L'approche C++ : le RAII

La réponse généralisée de C++ au problème de la gestion des ressources (et pas seulement au cas particulier de la gestion de la mémoire) s'appelle le RAII. Il s'agit de l'acronyme de Resource Acquisition Is Initialization, qui peut être traduit par acquisition des ressources au moment de l'initialisation. En réalité, il ne faut pas trop s'attacher au sens de cet acronyme, dans la mesure où il

ne traduit pas parfaitement le concept qu'il qualifie. La définition de la FAQ C++ (Lien28) dit : *Il s'agit d'un idiome de programmation consistant à manipuler une ressource quelconque (mémoire, fichier, mutex, connexion à une base de données, ...) au moyen d'une variable locale qui va acquérir cette ressource lors de son initialisation et la libérer lors de sa destruction.*

Cet idiome tire en fait profit d'une des particularités du langage C++ : la présence d'un destructeur de classe déterministe, autrement dit d'une fonction qui est *automatiquement* et *systématiquement* exécutée dès qu'un objet est détruit. Le principe du RAII est donc de se servir de cette parité entre le constructeur et le destructeur pour mettre en oeuvre, à la sauce C++ cette fois, le concept présenté plus haut : C'est celui qui a alloué un objet qui devrait le détruire.

Rappel : le seul cas où le destructeur d'un objet n'est pas appelé alors que son constructeur l'a été est celui où ce dernier a levé une exception. En effet, si un constructeur lève une exception, l'objet n'est pas considéré comme construit, et donc son destructeur n'est pas exécuté. Soyez donc vigilant sur ce point avec les ressources que vous allouez dans le constructeur et libérez dans le destructeur : vous risquez une fuite si une exception est levée peu après leur allocation alors que l'on se trouve toujours dans le constructeur.

Présenté autrement, le RAII consiste à matérialiser une ressource au moyen d'un objet qui acquiert cette ressource lors de son initialisation, et s'assure de sa bonne libération à sa destruction, même si le programmeur n'y a pas pensé. Le constructeur fait donc office de fonction Init() et le destructeur de fonction Free(). L'exemple qui suit met en oeuvre une classe Resource gérée traditionnellement et une autre classe ResourceRAII dont le design respecte les concepts du RAII.

```
class Ressource{
public:
    // Allocation des ressources : doit etre appelé
    // en premier, et une seule fois!
    bool Init();
    // Libération des ressources allouées : ne pas
    // oublier de l'appeler une fois terminé!
    void Free();

    void DoSomething();
};

class RessourceRAII{
public:
    RessourceRAII();
    ~RessourceRAII();

    void DoSomething();
};

void TestRessource(){
    Ressource r;
    if ( r.Init() ){
        try {
            r.DoSomething();
        } catch ( ... ) { // pseudo finally
            r.Free();
            throw; // relancer
        }
        r.Free();
    } else {
        // heu... que faire, quel est le problème ???
    }
}
```

```
void TestRessourceRAII(){
    RessourceRAII r; // exception levée si échec
    r.DoSomething();
}
```

Cet exemple illustre assez bien la très grande simplification d'utilisation et la robustesse qu'apporte le RAII. Car le principe du RAII veut aussi souvent que, si l'objet a été créé avec succès, alors il est directement utilisable. S'il ne parvient pas à s'initialiser, il peut lever une exception pour annuler sa construction. L'idée derrière ce comportement est : A quoi me servirait un objet qui n'a pas pu se construire et qui n'est pas conséquent pas utilisable ? Mais bien sûr, il existe des cas particuliers (comme std::ifstream par exemple).

Voyons maintenant comment solutionner notre problème initial de gestion des objets Resource en l'abordant sous ce nouvel angle.

5. Mise en oeuvre : premier essai

5.1. Présentation de shared_ptr

La solution la plus simple et la plus économique pour mettre en oeuvre le RAII est certainement de recourir à des pointeurs intelligents (Lien29). Les pointeurs intelligents sont l'exemple par excellence d'application concrète de cet idiome. Ils le marient avec le concept de généricité des templates, ce qui permet une mise en oeuvre du RAII rapide et à moindre frais. En fait, les pointeurs intelligents sont les compagnons indispensables de tout programmeur C++ sérieux.

Il en existe de nombreuses et diverses implémentations. J'ai choisi shared_ptr, historiquement originaire de la bibliothèque boost (Lien30) et maintenant membre du TR1 (Lien31). Cela signifie que shared_ptr est en train d'être adopté au sein de la norme du langage et sera donc à terme disponible en standard avec la plupart des compilateurs.

En attendant ce jour futur, force est de constater que l'état actuel des choses (Juillet 2007) est différent. Seul GCC propose std::tr1::shared_ptr en standard, et encore, pas sous toutes les plateformes. Mais les choses ne sont pas si négatives que cela non plus, vu qu'il existe la très mature implémentation boost::shared_ptr, et qu'il existe aussi Boost.TR1 (Lien32) qui fournit des fichiers d'en-tête redéfinissant ce type (ainsi que d'autres) au sein de l'espace référentiel std::tr1. En plus clair, il est parfaitement possible d'utiliser le type std::tr1::shared_ptr tel qu'il est défini dans le Technical Report 1 avec la plupart des compilateurs à condition d'installer Boost.TR1. A ce sujet, vous pouvez lire *Installer et utiliser Boost/Boost.TR1 avec Visual C++* (Lien33).

5.2. Premier essai

Reprenons l'exemple initial et adaptons-le pour utiliser std::tr1::shared_ptr:

ResourceManager.h

```
#include <tr1/memory>

class Resource;
typedef std::tr1::shared_ptr<Resource> ResourcePtr;

class ResourceManager {
public:
    // Définit le nombre maximal de Resource
utilisables simultanément
```

```
    ResourceManager( int MaxNbInstances );

    // Renvoie une nouvelle Resource s'il y en a moins
que
    // MaxNbInstances en cours d'utilisation, NULL
sinon.
    ResourcePtr New();

private:
    // Nombre d'instances en cours d'utilisation
    int NbInstances;
    // Nombre maximal d'instances autorisées
    const int MaxNbInstances;
};
```

Vis à vis du programmeur client, notre ResourceManager est on ne peut plus simple et fiable d'utilisation : on alloue une nouvelle ressource via la fonction ResourceManager::New(), on l'utilise sans se poser de question, et une fois qu'on en a terminé avec elle, ResourceManager est automatiquement informé et mis à jour qu'une ressource a été libérée.

Ceci est séduisant, mais il reste encore à coder le ResourceManager est automatiquement informé et mis à jour.

La première solution qui vient à l'esprit est de modifier le comportement de Resource pour que la classe informe son gestionnaire qu'elle a été détruite.

Resource.h

```
class ResourceManager;

class Resource{
public:
    // Pointeur NULL = pas rattaché à un
ResourceManager
    Resource( ResourceManager* = 0 );

    // Avertit le ResourceManager attaché de sa
destruction
    ~Resource();

private:
    // ResourceManager attaché, peut etre NULL
    ResourceManager *Manager;
};
```

ResourceManager.h

```
#include <tr1/memory>

class Resource;
typedef std::tr1::shared_ptr<Resource> ResourcePtr;

class ResourceManager{
public:
    // Définit le nombre maximal de Resource
utilisables simultanément
    ResourceManager( int MaxNbInstances );

    // Renvoie une nouvelle Resource s'il y en a moins
que
    // MaxNbInstances en cours d'utilisation, NULL
sinon.
    ResourcePtr New();

private:
    friend class Resource;
    // Appelé par les objets Resources lors de leur
destruction
    void ResourceDestroyed( Resource* );

private:
    // Nombre d'instances en cours d'utilisation
    int NbInstances;
    // Nombre maximal d'instances autorisées
    const int MaxNbInstances;
};
```

Resource.cpp

```
#include "Resource.h"
#include "ResourceManager.h"

Resource::Resource( ResourceManager *Mgr ):
    Manager( Mgr )
    {}

Resource::~Resource(){
    if ( this->Manager ) {
        this->Manager->ResourceDestroyed( this );
    }
}
```

ResourceManager.cpp

```
#include "ResourceManager.h"
#include "Resource.h"

ResourceManager::ResourceManager( int Max ) :
    NbInstances( 0 ),
    MaxNbInstances( Max )
    {}

ResourcePtr ResourceManager::New(){
    if ( NbInstances < MaxNbInstances ){
        ResourcePtr r( new Resource() );
        // On incrémente NbInstances _après_ avoir
alloué
        // la ressource avec succès, afin de laisser
notre
        // Manager dans un état cohérent si une
exception
        // est levée lors de la création d'un objet
Resource
        ++this->NbInstances;
        return r;
    }
    return ResourcePtr(); // NULL
}

void ResourceManager::ResourceDestroyed( Resource* ){
    --NbInstances;
}
```

5.3. Nouveaux problèmes soulevés

Cette solution fonctionne, mais pose de nouveaux problèmes :

- La modification de la classe Resource. Ceci n'est pas toujours possible. Il pourrait très bien s'agir d'une classe issue d'une bibliothèque tierce. Dans ce cas, il faudrait alors développer une classe proxy supplémentaire, ce qui est typiquement une contrainte dont on aime se passer.
- La référence croisée entre Resource et ResourceManager. Souvent signe d'une mauvaise conception, les références croisées augmentent le couplage entre les classes concernées, ce qui complique leur maintenance. Et dans ce cas précis, elle pose le problème supplémentaire que, conceptuellement, elle n'a pas lieu d'être. En effet, si l'on se place du point de vue de la classe Resource, on a que faire de savoir si on est géré via un ResourceManager ou pas. Et pourtant, cette information qui ne nous concerne pas vient parasiter

notre design, alors qu'elle relève du détail d'implémentation de ResourceManager!

- L'utilisation du mot-clé **friend** est lui aussi souvent un indice d'une mauvaise conception. Le mot-clé en lui même est une bonne chose puisqu'il permet de renforcer l'encapsulation de ResourceManager en rendant sa fonction membre ResourceDestroyed() **private**. Sans lui, nous aurions du la rendre **public**, ce qui aurait été pire. Mais il n'empêche que dans notre cas, il traduit le besoin qu'a la classe que nous gérons d'assurer elle-même sa propre gestion, alors que c'est l'unique raison d'être de ResourceManager.
- Cette solution est difficilement généralisable, sous forme d'une classe ResourceManager template par exemple.

6. Nouvel essai : explorons les possibilités de shared_ptr

Notre nouvel objectif est donc de parvenir à appliquer le RAII sans avoir à modifier Resource et sans introduire de référence croisée vis à vis de ResourceManager. Comment faire ?

Si l'on réfléchit un instant, notre problématique est d'être informé de la destruction d'un objet Resource que l'on a nous même alloué un peu plus tôt. Demander à cet objet de nous prévenir quand cela se produit était une première possibilité, mais nous avons vu qu'elle comportait de nombreuses lacunes.

Quand on se retrouve confronté à une situation de dépendance circulaire, la solution pour s'en sortir consiste souvent à introduire un troisième acteur. Et justement, nous disposons d'un troisième acteur : shared_ptr. Son utilisation est presque passée inaperçue car discrète, mais il n'en demeure pas moins que nous avons introduit un proxy sur la classe Resource. Et l'avantage d'utiliser shared_ptr est que cette classe dispose de nombreuses fonctionnalités évoluées.

En particulier, shared_ptr dispose d'un constructeur très intéressant, acceptant en second paramètre un deallocator :

```
template<class Y, class D>
shared_ptr(Y * p, D d);
```

Le rôle premier d'un deallocator est de libérer la mémoire attribuée à l'instance de l'objet dont le pointeur est encapsulé. L'implémentation par défaut effectue un simple appel à delete. Mais comme le précise la documentation de shared_ptr (Lien34), le concept de deallocator ouvre la voie à d'autres types d'utilisations :

Custom deallocators allow a factory function returning a shared_ptr to insulate the user from its memory allocation strategy. Since the deallocator is not part of the type, changing the allocation strategy does not break source or binary compatibility, and does not require a client recompilation. For example, a "no-op" deallocator is useful when returning a shared_ptr to a statically allocated object, and other variations allow a shared_ptr to be used as a wrapper for another smart pointer, easing interoperability.

Dans notre cas, nous allons aller un peu plus loin et utiliser le deallocator comme fonction callback exécutée au moment de la destruction d'un objet Resource, un peu comme un second destructeur en somme!

ResourceManager.h

```
#include <tr1/memory>

class Resource;

typedef std::tr1::shared_ptr<Resource> ResourcePtr;

class ResourceManager{
public:
    // Définit le nombre maximal de Resource
utilisables simultanément
    ResourceManager( int MaxNbInstances );

    // Renvoie une nouvelle Resource s'il y en a moins
que
    // MaxNbInstances en cours d'utilisation, NULL
sinon.
    ResourcePtr New();

private:
    class ResourceDeleter; // nested class
    friend ResourceDeleter;

private:
    // Nombre d'instances en cours d'utilisation
    int NbInstances;
    // Nombre maximal d'instances autorisées
    const int MaxNbInstances;
};
```

ResourceManager.cpp

```
#include "ResourceManager.h"
#include "Resource.h"

// Classe privée chargée de détruire les objets
Resource et
// de mettre à jour le compteur NbInstances de
ResourceManager
// (Utilisée comme deleter personnalisé de shared_ptr)
class ResourceManager::ResourceDeleter{
public:
    ResourceDeleter( ResourceManager *Mgr ):
        Manager( Mgr ){
    }

    void operator()( Resource *Res ){
        --(this->Manager->NbInstances);
        delete Res;
    }

private:
    ResourceManager *Manager;
};

ResourceManager::ResourceManager( int Max ) :
    NbInstances( 0 ),
    MaxNbInstances( Max )
    {}

ResourcePtr ResourceManager::New(){
    if ( NbInstances < MaxNbInstances ){
        ResourcePtr r( new Resource(), ResourceDeleter(
this ) );
        // On incrémente NbInstances _après_ avoir
alloué
        // la ressource avec succès, afin de laisser
notre
        // Manager dans un état cohérent si une
exception
        // est levée lors de la création d'un objet
Resource
        ++this->NbInstances;
        return r;
    }
    return ResourcePtr(); // NULL
}
```

Elégant n'est-ce pas ? Il n'est plus nécessaire de modifier Resource (qui peut donc être une classe issue d'un code dont vous n'avez pas le contrôle), et la référence croisée a disparu. Certes, il reste l'utilisation du mot-clé friend, mais ce dernier porte cette fois sur une classe imbriquée privée de ResourceManager, et n'est donc pas choquante. ResourceDeleter fait en effet partie de l'implémentation de ResourceManager, et les deux classes évolueront donc ensemble de manière naturelle.

7. Version finale

Cependant, il y a des personnes comme moi qui sont vraiment récalcitrantes à utiliser le mot clé **friend**. J'interprète en effet sa présence comme le signe d'un possible défaut dont on essaye de limiter l'étendue, au lieu de le corriger. Et dans ce cas, le défaut est de faire figurer dans l'interface de ResourceManager un détail d'implémentation, appelé ResourceDeleter. C'est un défaut discutable, dans la mesure où il s'agit de l'interface privée de la classe, et que le C++ impose de la rendre visible dans le fichier d'en-tête. Mais en ce qui me concerne, j'aime bien réduire cette interface privée exposée au minimum, dans la mesure du possible bien sûr. Car si l'utilisateur n'a pas le droit ni la possibilité d'utiliser ResourceDeleter, pourquoi l'informer de son existence ?

Et justement, dans ce cas, il est possible de supprimer cette information du fichier d'en-tête en cloisonnant ResourceManager dans un espace de nommage anonyme du fichier d'implémentation :

ResourceManager.h

```
#include <tr1/memory>
#include <boost/utility.hpp>

class Resource;
typedef std::tr1::shared_ptr<Resource> ResourcePtr;

class ResourceManager : public boost::noncopyable{
public:
    // Définit le nombre maximal de Resource
utilisables simultanément
    ResourceManager( int MaxNbInstances );

    // Renvoie une nouvelle Resource s'il y en a moins
que
    // MaxNbInstances en cours d'utilisation, NULL
sinon.
    ResourcePtr New();

    // Renvoie le nombre d'objets Resource en cours
d'utilisation
    int GetNbInstances() const;

private:
    // Nombre d'instances en cours d'utilisation
    int NbInstances;
    // Nombre maximal d'instances autorisées
    const int MaxNbInstances;
};
```

ResourceManager.cpp

```
#include "ResourceManager.h"
#include "Resource.h"

namespace // anonyme{
    // Classe chargée de détruire les objets Resource
et
    // de mettre à jour le compteur NbInstances de
ResourceManager
    // (Utilisée comme deleter personnalisé de
shared_ptr)
    class ResourceDeleter{
    public:
        ResourceDeleter( int *MgrNbInstances ):
            Manager_NbInstances( MgrNbInstances )
        {}

        void operator()( Resource *Res ){
            --(*this->Manager_NbInstances);
            delete Res;
        }

    private:
        // pointeur sur ResourceManager::NbInstances
        int *Manager_NbInstances;
    };
}

ResourceManager::ResourceManager( int Max ) :
    NbInstances( 0 ),
    MaxNbInstances( Max )
    {}

ResourcePtr ResourceManager::New(){
    if ( NbInstances < MaxNbInstances ){
        ResourcePtr r( new Resource(), ResourceDeleter(
&this->NbInstances ) );
        // On incrémente NbInstances _après_ avoir
alloué
        // la ressource avec succès, afin de laisser
notre
        // Manager dans un état cohérent si une
exception
        // est levée lors de la création d'un objet
Resource
        ++this->NbInstances;
        return r;
    }
    return ResourcePtr(); // NULL
}

int ResourceManager::GetNbInstances() const{
    return this->NbInstances;
}
```

Pour donner accès à une donnée membre privée de ResourceManager sans introduire friend, je suis obligé de recourir à un pointeur. Je ne suis en général pas très adepte de ce genre d'acrobatie visant en fait à contourner le contrôle d'accès du compilateur. Mais appliquée à ce cas précis, elle me semble acceptable.

Certains auraient peut-être utilisé une référence à la place d'un pointeur. J'ai choisi un pointeur car je trouve qu'il rend plus explicite le fait que la classe ResourceDeleter va modifier l'int reçu en paramètre. Mais libre à vous d'utiliser une référence si vous préférez.

Enfin, vous remarquerez l'ajout d'une fonction membre GetNbInstances() ainsi que l'inclusion de boost/utility.hpp dans le but de faire dériver ResourceManager de boost::noncopyable (Lien35). Ceci permet de protéger notre gestionnaire d'une copie accidentelle, et d'indiquer de manière plus parlante que cette classe n'est pas faite pour copiée (le moyen habituel d'opérer est de déclarer privés, sans les implémenter, le constructeur par recopie ainsi que l'opérateur d'affectation). Cela permet aussi, avec Visual C++, de faire disparaître l'avertissement 4512 (niveau 4)(Lien36) : 'ResourceManager' : l'opérateur d'assignation n'a pas pu être généré.

Retrouvez l'article entier d'Aurélien Regat-Barrel en ligne : Lien37

Foundations of GTK+ Development

GTK+ is one of the most influential graphical toolkits for the Linux operating system. It is the technology upon which the GNOME and XFCE desktop environments are based, and it's crucial to have clear understanding of its complexities to build even a simple Linux desktop application. Foundations of GTK+ Development guides you through these complexities, laying the foundation that will allow you to cross from novice to professional.

Foundations of GTK+ Development is aimed at C programmers and presents numerous real-life examples that you can immediately put to use in your projects. Some familiarity with C programming is assumed, as the book delves into new topics from the beginning. Topics like object inheritance are covered early on to allow for complete understanding of code examples later. And the provided examples are real-life situations that can help you get a head start on your own applications.

Critique du livre par la rédaction (gege2061)

Les livres traitant de GTK+ se font rares et ceux basés sur GTK+ 2.0 sont, à ma connaissance, inexistant. Plus précisement étaient inexistant, puisque ce nouveau livre nous proposent de maitriser l'une des plus puissantes bibliothèques multi-plateforme !

Malgrès la faible épaisseur du livre (oui 600 pages pour couvrir tout GTK+ c'est pas énorme), l'auteur nous propose un large aperçu de GTK+ : les widgets de base sont rapidement abordés, les widgets plus élaborés (text view, tree view par exemple) occupent un chapitre entier. Mais l'auteur ne s'arrête à GTK+, il y a également un chapitre sur la glib mais aussi sur glade.

Chaque notion est acompagnée d'un exemple relativement court mais complet. Comme ses exemples ont un interêt relativement limité, le dernier chapitre présente 5 applications complètes (du gestionnaire de fichier au jeux du pendu) dont les codes sources sont disponibles dur le site du livre : http://www.gtkbook.com/ (vous y trouverai même un tutoriel sur l'utilisation du parser GKeyFile de la glib).

Pour compléter le tout, vous trouverai des exercices à la fin de chaque chapitre avec leur correction.

Même si le livre s'adresse aux débutants, tous les utilisateurs peuvent y trouver leur compte grâce aux nombreux exemples et parce qu'il y a toujours une fonctionnalité fort utile à côté de laquelle on est passé.

Ma conclusion sur ce livre sera très rapide puisque le seul point négatif est que ce livre est en anglais. on objectif, c'est grâce à lui que j'ai réussi mon examen SCJP 5.

Retrouvez ce livre sur la rubrique GTK : Lien38

Les derniers tutoriels et articles

Sécurité granulaire

Comment gérer l'accès partiel aux données d'une table ?

1. Introduction

La sécurité granulaire (Fine Grained Access Control) nous permet de gérer l'accès d'un sous-ensemble
d'enregistrements d'une table (l'accès partiel aux données d'une table).

La sécurité granulaire figure aussi sous d'autres noms dans la documentation d'oracle, dont on peut citer :

- **La sécurité de niveau rangée** (RLS).
- La politique ou règle de sécurité (policy) de **bases de données privées virtuelles** (VPD).

Exemple :
Dans une table dossier (no_dos,dt_dos,type_dos(secret,normal))

- Les dossiers secrets sont accessibles uniquement par les managers(les usagers qui ont le rôle sec_manager).
- Les employés ont seulement le droit de voir les dossiers normaux(les usagers qui ont le rôle sec_employe).

2. Les composants de la sécurité granulaire

La sécurité granulaire est constituée de trois éléments :

2.1. Le contexte et son package

2.1.1. Le contexte (context)

Le contexte est un regroupement de variables, sa synthaxe est la suivante :
Create or replace context nom_context **using** nom_proc_ou_pkg

```
SQL> create or replace context scott_dossier using
pkg_dossier_context;
```

2.1.2. Le package de contexte

Un contexte est toujours lié à un package et c'est le seul qui a le droit d'initialiser les variables de ce contexte.

Remarque:
Si on utilise un package ou une procédure pour initialiser les variables de context
autre que le package ou la procédure déclaré au moment de la création du context,
il nous retourne l'erreur suivante :

```
Ora-01031 : insufficient privileges
Ora-06512 : at « SYS.DBMS_SESSION », at line ..
ora-65512  : at line 2
```

2.2. Les politiques ou règles (policies) et son package

2.2.1. Politique ou règle (policy)

Nous permet d'attacher la sécurité à une table, et déterminer les opérations DML(select, insert, update, delete) concernées par la sécurité pour cette même table.
Quand on élimine une table, la politique ou règle (policy) de cette table disparaît automatiquement. On peut avoir plusieurs politiques ou règles (polices) par table.

```
SQL> begin
  2       dbms_rls.add_policy(
  3                 object_schema => 'SCOTT',
  4                 object_name => 'DOSSIERS',
  5                 policy_name =>
'SCOTT_DOSSIERS_POLICY',
  6                 function_schema => 'SCOTT',
  7                 policy_function =>
'pkg_dossier_sec.dossier_predicate',
  8                 statement_types => 'select, insert,
update, delete',
  9                 update_check => TRUE,
 10                enable => TRUE,
 11                static_policy => FALSE);
 12    end;
 13  /
PL/SQL procedure successfully completed.
```

Remarque:
Le statement_types => 'select, insert, update, delete' qui nous permet
de déterminer les opérations DML qui sont concernées par la sécurité.

2.2.2. Package de sécurité

Permet d'ajouter une clause de restriction à la requête originale, on l'appelle prédicat.

2.3. Trigger on_logon

Nous permet d'initialiser les variables du contexte, il se déclenche juste après la création d'une session Oracle.

- S'exécute à chaque connexion.
- Les variables d'un contexte sont disponibles et peuvent être utilisées dans un module, une procédure...etc
- Attention car la soumission des rapports déclenche une nouvelle session.

```
create or replace trigger scott_logon_trigger
     after logon on database
     declare
     NB VARCHAR2(30) ;
```

```
     begin
        select granted_role
           into nb
           from dba_role_privs    where grantee='SCOTT'
and
           granted_role='SEC_MANAGER' ;
      pkg_dossier_ context.set_manager;
     exception
      when no_data_found then
      pkg_dossier_context.set_employe;
      end;
   /
```

La politique ou règle (policy) est le plus important composant à la sécurité granulaire.
Les autres éléments rendent cette sécurité plus efficace et le système n'est pas ralenti.

Retrouvez l'article de Salim Chelabi en ligne avec un exemple de sécurité granulaire : Lien39

Réplication Oracle avec Streams

1. Introduction

Par le passé, les systèmes de réplication qu'offraient Oracle n'ont jamais fait l'unanimités. Basés sur des snapshots de tables, peu performants, ils ont souvent été ignorés au profit d'autres systèmes de réplication hétérogènes d'autres éditeurs.

Dès la version 9i, Oracle a décidé de palier à cette carence en réécrivant totalement sa réplication, se basant sur l'expérience de transfuges d'autres éditeurs. C'est ainsi que Streams a vu le jour.

les débuts de Streams ont été douloureux, et il n'est d'ailleurs pas recommandé de l'utiliser en production avec une masse importante de données à répliquer en version pré-10.2. Dès la version 10.2, Streams commence par contre à devenir un outil de réplication transactionnel intéressant.

Cet article à pour but de vous démontrer les fonctionnalités basiques de Streams. Afin de ne pas sombrer dans la traduction stupide d'articles Oracle, je me propose de vous démontrer la mise en place d'une configuration Streams particulière : Downstreams (Archieved-Log Downstreams Capture).

Principaux traits de Downstreams

- Quasi aucun impact de performance sur la base source, la lecture se faisant via des ficher d'archivelogs, et tous les processus de Streams étant localisés sur l'instance cible
- Quasi aucun impact sur la source si la cible vient à tomber en panne
- Réplication à flux semi tendu puisque les données n'arrivent qu'au rythme des archivelogs (il faut donc attendre ou générer des switchlogs)
- Architecture intéressante pour une réplication d'un environnement OLTP à l'ODS d'un environnement décisionnel
- Possibilité de modifier les règles de réplication pour, par exemple, transférer les données d'une table source dans un schéma et une table dont les noms sont différents sur la cible
- Possibilité (dès 10.2) de déterminer des règles négatives afin d'améliorer les performances du processus capture.

Nous partirons du postulat que nous allons répliquer des données d'un serveur source nommé ORAOLTP à un serveur cible nommé ORADSS.

En résumé, voici ce qui va se passer

1. Des utilisateurs modifient des enregistrements sur ORAOLTP
2. Les blocs modifiés passent au travers du redo log
3. Lorsque le redo log est plein ou qu'un switch redo log est activé, le redo suivant est activé.
4. Le redo traité est journalisé, soit donc copié comme archivelog par l'archiver
5. Downstreams oblige en plus l'archiver à copié une copie de ce fichier sur le site distant de ORADWH
6. Dès ce point s'arrête la charge sur ORAOLTP, et commence le travail de ORADSS
7. Le processus de capture d'ORADWH détecte l'arrivée d'un nouvel archivelog distant
8. Il le lit intégralement et en retire les modifications apportées sur les tables qui sont marquées comme répliquées
9. Il crée un enregistrement dans une file d'attente et l'envoie au propagateur
10. Le propagateur applique au besoin des transformations et l'envoie à l'applicateur (applyer)
11. L'applicateur exécute la modification sur la base ORADWH

II. Configuration de l'instance source

Afin de permettre le bon fonctionnement de Streams un certain nombre de paramètres de l'instance source doivent être configurés avec des valeurs spécifiques :

Paramètre	Valeur
log_archive_config	SEND
log_archive_dest_2	SERVICE=ORADWH.DEVELOPPEZ.COM ARCH OPTIONAL NOREGISTER TEMPLATE=/CheminSurServeurCible/ORAOLTP_%t_%s_%r.arc
log_archive_dest_state_2	Enable

Création d'un super-utilisateur STREAMSADM

```
CONNECT SYS/******@ORAOLTP AS SYSDBA
CREATE USER STREAMSADM IDENTIFIED BY ******** ACCOUNT
UNLOCK ;
GRANT DBA TO STREAMSADM ;
GRANT RESOURCES TO STREAMSADM ;
```

3. Configuration de l'instance cible

Afin de permettre le bon fonctionnement de Streams un certain nombre de paramètres de l'instance cible doivent être configurés avec des valeurs spécifiques :

Paramètre	Valeur
log_archive_config	RECEIVE

Création d'un super-utilisateur STREAMSADM et de son schéma, en tant que SYS

En configuration Downstreams, les mots de passe de SYS doivent être obligatoirement similaires sur ORADWH que sur ORAOLTP. Par convenance, nous agirons de même en ce qui concerne l'utilisateur STREAMADM : cela simplifie certaines tâches administratives.

```
CONNECT SYS/******@ORADWH AS SYSDBA
CREATE TABLESPACE STREAMS_D01 DATAFILE
'/VotreLocalisation/STREAMSADM_D01.dat' SIZE=200M
AUTOEXTEND ON NEXT 100M MAXSIZE 2G ;
CREATE USER STREAMSADM IDENTIFIED BY ********  DEFAULT
TABLESPACE STREAMS_D01 ACCOUNT UNLOCK;
GRANT DBA TO STREAMSADM ;
GRANT RESOURCES TO STREAMSADM ;
GRANT EXECUTE ON DBMS_FLASHBACK, dbms_streams_adm,
dbms_apply_adm, dbms_streams_adm TO STREAMSADM ;
BEGIN
  DBMS_STREAMS_AUTH.GRANT_ADMIN_PRIVILEGE(
    grantee          => 'STREAMSADM',
    grant_privileges => true);
END;
/
```

Bien que l'environnement est en mode Downstreams, il est quand même nécessaire d'avoir un database link entre la base de données cible et la base de données source. Ce database link est à créer sur la base de données cible. Il permet de faciliter l'opération d'enregistrement de nouvelles tables dans le flux de réplication Streams. Le database link appartient à l'administrateur Streams (STREAMSADM).

```
CONNECT STREAMSADM/*******@ORADWH
-- création
CREATE DATABASE LINK ORAOLTP CONNECT TO CURRENT_USER
USING 'ORAOLTP.DEVELOPPEZ.COM';

-- Test
SELECT  *  FROM DUAL@ORAOLTP ;
```

Afin de permettre la configuration des processus de capture et d'application il est nécessaire de créer 2 queues Oracle de type Anydata sur la base de données cible. Cette création s'effectue au moyen de la procédure DBMS_STREAMS_ADM.SET_UP_QUEUE

```
PROMPT Création de la queue de capture
begin
    dbms_streams_adm.set_up_queue(
       queue_table  => 'streamsadm.stream_queue_cpt',
       queue_name   => 'stream_queue_cpt',
       queue_user   => 'streamsadm');
end;
/

-- Test
SELECT * FROM streamsadm.stream_queue_cpt ;
```

Selon la même méthode, création d'une queue d'application nommée stream_queue_appl et appartenant à l'administrateur Streams (STREAMSADM).

```
PROMPT Création de la queue d'application

begin
     dbms_streams_adm.set_up_queue(
       queue_table  => 'streamsadm.stream_queue_appl',
       queue_name   => 'stream_queue_appl',
       queue_user   => 'streamsadm');
end;
/

-- Test
SELECT * FROM streamsadm.stream_queue_appl ;
```

La dernière étape à réaliser avant de pouvoir ajouter une table dans le processus de réplication Streams et la création du processus de Capture sur la base cible. Cette création s'effectue au moyen du package DBMS_CAPTURE_ADM.CREATE_CAPTURE.

```
BEGIN
  DBMS_CAPTURE_ADM.CREATE_CAPTURE(
    queue_name          =>
'streamsadm.stream_queue_cpt',
    capture_name        => 'strm_capture',
    rule_set_name       => NULL,
    start_scn           => NULL,
    source_database     => 'ORAOLTP',
    use_database_link   => true,
    first_scn           => NULL,
    logfile_assignment => 'implicit');
END;
/

-- Test

SELECT CAPTURE_NAME, STATUS, STATUS_CHANGE_TIME FROM
DBA_CAPTURE ;
```

4. Heardbeat : la première table répliquée à mettre en place

Afin d'être certain que notre système de réplication fonctionne, il est d'usage de créer une table de test sur laquelle on opère un update chaque 5 minutes, via un cron ou un job Oracle. Cette table permet alors de calculer la latence de la réplication.

Première table à répliquer

```
DECLARE
   cStreamUser CONSTANT VARCHAR2(30)           :=
'STREAMSADM';
   cStreamPropName CONSTANT VARCHAR2(30)       :=
'STRM_PROP';
   cStreamApplyName CONSTANT VARCHAR2(30)      :=
'STRM_APPLY';
   cStreamCaptureName CONSTANT VARCHAR2(30)    :=
'STRM_CAPTURE';
   cSourceQueueName CONSTANT VARCHAR2(30)      :=
cStreamUser || '.' || 'STREAM_QUEUE_CPT';
   cDestinationQueueName CONSTANT VARCHAR2(30) :=
cStreamUser || '.' || 'STREAM_QUEUE_APPL';

   pOwner VARCHAR2(30) := 'STREAM';
   pTableName VARCHAR2(30) := 'STREAM_LASTCOMMIT';
   pSourceInstance VARCHAR2(30) :=
'ORAOLTP.DEVELOPPEZ.COM';
   iScn NUMBER := 0 ;
   vSql varchar2(500) ;

 BEGIN

 DBMS_STREAMS_ADM.ADD_TABLE_RULES(
                                table_name
       =>  Upper(pOwner)||'.'||Upper(pTableName),
                                streams_type         =>
'capture',
                                streams_name         =>
cStreamCaptureName,
                                queue_name
       =>  cSourceQueueName,
                                include_dml          =>
true,
```

```
                                include_ddl           =>
false,
                                include_tagged_lcr    =>
false,
                                source_database       =>
pSourceInstance,
                                inclusion_rule
       =>  true);

 -- Création de la règle de propagation

dbms_streams_adm.add_table_propagation_rules(
                        table_name                    =>
Upper(pOwner)||'.'||Upper(pTableName),
                        streams_name          =>
cStreamPropName,
                    source_queue_name         =>
cSourceQueueName,
                    destination_queue_name    =>
cDestinationQueueName,
                    include_dml                   =>
TRUE,
                    include_ddl                   =>
FALSE,
                    include_tagged_lcr        => FALSE,
                    source_database           =>
pSourceInstance);

dbms_streams_adm.add_table_rules(
                table_name                        =>
pOwner||'.'||pTableName,
                streams_type                      =>
'APPLY',
                streams_name                      =>
cStreamApplyName,
                queue_name                        =>
cDestinationQueueName,
                include_dml                       =>
true,
                include_ddl                       =>
FALSE,
                include_tagged_lcr                =>
false,
                inclusion_rule                    =>
true,
                source_database                   =>
pSourceInstance);

vSql := 'SELECT
DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER()  into iSCN
FROM DUAL@'||pSourceInstance ;
execute immediate (vSql) ;

vSql := 'truncate table '|| pOwner||'.'||pTableName ;
execute immediate (vSql) ;

vSql := 'insert into '|| pOwner||'.'||pTableName
           ||' select * from '||pOwner||'.'||
pTableName||'@'||pSourceInstance ;
execute immediate (vSql) ;

dbms_apply_adm.set_table_instantiation_scn (
       source_object_name                 =>
Upper(pOwner)||'.'||Upper(pTableName),
       source_database_name          =>
pSourceInstance,
                instantiation_scn             => iSCN
                ) ;
END;
/
```

5. Contrôle de l'état de Streams

Outre la Grid Control qui permet une administration assez aisée de Streams, voici quelques requêtes intéressantes à encapsuler dans vos scripts de contrôle.

Statuts des divers processus

```
SELECT 'CAPTURE', C.CAPTURE_NAME, C.STATUS,
C.STATUS_CHANGE_TIME
FROM DBA_CAPTURE C
UNION ALL
SELECT 'APPLY', a.APPLY_NAME, a.STATUS,
a.STATUS_CHANGE_TIME
from DBA_APPLY a
```

Erreurs apparaissant lors de l'application

```
SELECT * from DBA_APPLY_ERROR ;
```

Les transactions bloquantes sont stockées comme des messages. Il n'est donc pas aisé, en cas de blocage, de détecter quelle transaction bloque et pourquoi. Oracle propose à cet effet la fonction print_transaction() qui affiche, pour un no de transaction donné par la table DBA_APPLY_ERROR.

Je vous conseille donc vivement de suivre la documentation Oracle pour créer cette fonction fort utile.

Utilisation de la fonction print_transaction()

```
SET SERVEROUTPUT ON SIZE 1000000

EXEC print_transaction('1.12.3455')
```

Pas de SQL dans ce que vous obtiendrez : on décortique assez aisément le message en lisant les nouvelles et anciennes valeurs des objets. Relevez rapidement le nombre de modifications par message et le type du traitement apparaissant dans l'entête.

Monitoring des tables répliquées

```
select  substr(rule_condition,
instr(rule_condition,'''')+1,
instr(substr(rule_condition,
instr(rule_condition,'''')+1),'''')-1) SCHEMAS,
        substr(rule_condition, instr(rule_condition,
'get_object_name()')+21,
        instr(rule_condition, ')) and
:dml.is_null_tag()') - instr(rule_condition,
'get_object_name()')-22) TABLES,
        source_database
from sys.STREAMS$_RULES
where streams_name='STRM_CAPTURE';
```

6. Nettoyage de Streams

Il peut être nécessaire parfois de redémarrer complètement l'environnement Streams depuis zéro. Pour ce faire il est nécessaire de supprimer toutes les tables du flux de réplication Streams.

Selon la documentation Oracle le nettoyage de l'environnement Streams peut être fait avec l'exécution de la procédure DBMS_STREAMS_ADM.REMOVE_STREAMS_CONFIGUR ATION. Malheureusement ce n'est pas le cas, en plus de ce package il est nécessaire de nettoyer les règles créées lors de l'ajout de table et de supprimer les queues d'Apply et de Capture.

```
BEGIN
DBMS_STREAMS_ADM.REMOVE_STREAMS_CONFIGURATION ;

FOR maqueue IN (select name from dba_queues where owner
= 'STREAMSADM')
LOOP
    DBMS_STREAMS_ADM.REMOVE_QUEUE(maqueue.name,TRUE);
END LOOP;
```

```
FOR maregle IN (select rule_set_name from dba_rule_sets
where rule_set_owner = 'STREAMSADM')
LOOP
   dbms_rule_adm.drop_rule_set(maregle.rule_set_name,TR
UE);
END LOOP;
END ;
/
```

7. Conséquences d'un cluster

Dans le cas d'un serveur source en cluster, soit donc avec plusieurs instances gérant une même base, la logique est la même, bien que chacune des instances gère ses propres archivelogs.

Les archivelogs sont gérées par chacune des instances et copiées de manière asynchrone sur la cible.

Le seul impact que le RAC génère est la latence de la réplication. Quel que soit l'ordre d'arrivée des archivelogs, c'est la séquence du SCN, global à la base et à toutes les instances sources qui déterminera l'ordre de lecture des archivelogs par le processus de capture.

Il convient donc, si les quelques instances sources n'ont pas le même rythme de génération des archivelogs (souvent le cas lorsqu'une instance et plus fortement sollicitée), de forcer via job Oracle un switch logfile à intervalle régulier sur les 2 instances.

Retrouvez l'article de Fabien Celaia en ligne : Lien40

Les livres SGBD

SQL Server 2005 : Entraînez-vous à administrer une base de données

Avec ce livre, le lecteur pourra s'entraîner sur les différentes opérations que peut être amené à réaliser un administrateur SQL Server 2005. Les exercices proposés couvrent l''installation, la gestion de l'espace disque, la sécurité, la sauvegarde, la restauration et permettent aussi de mettre en pratique certains apports intéressants de SQL Server 2005 comme le partitionnement de table, le service de notification, Service Broker ou bien encore le transfert de données avec SSIS.

Suivant le contexte les opérations sont réalisées de façon graphique avec SQL Server Management Studio et/ou sous forme de script Transact SQL.

Critique du livre par la rédaction (Grégory Dumas)

Je tiens, tout d'abord, à préciser un élément important, qui ne saute pas forcément aux yeux lorsque l'on arrive sur les pages des différents libraires en ligne : ce livre fait parti de la collection "Les TP informatiques" et, à ce titre, il n'y a pas tromperie sur la marchandise : il s'agit bien d'une série de travaux pratiques.

Le livre se divise en deux parties : les énoncés, les corrigés.
Pour chaque thême, une série de questions/réponses est posée pour vérifier si les pré-requis nécessaires à la bonne réalisation des TPs sont connus.

Ensuite la série de TPs suit. Chaque énoncé de TP est constitué d'un titre qui situe l'objet de l'exercice, d'une durée conseillée pour réaliser l'exercice, de l'énoncé en lui même et enfin quelques fois d'indices afin d'orienter la réponse possible. De plus, sur le site de l'éditeur, on peut retrouver des ressources qui peuvent être utiles à la bonne réalisation du TP.
Dans les corrigés, on retouve les réponses aux Questions/Réponses des pré-requis. Enfin suivent les corrigés des TPS à proprement parler.
Ces corrigés sont détaillés, aggrémentés de nombreuses captures d'écrans permettant au lecteur de suivre facilement les différentes étapes de la correction.
On pourra cependant regretter que les questions ne soient pas répéter dans les corrigés obligeant le lecteur à naviguer sans cesse entre les énoncés et les corrigés.

Ce livre s'adresse selon moi à un public que je qualifierai d'intermédiaire. En effet, les personnes expérimentés trouveront peu de motifs d'interêt; de nombreuses tâches décrites sont relativement courantes pour être déjà maitrisées. D'autres part, les débutants pourront apprendre à réaliser des tâches sans néanmoins en connaître ni le but, ni l'intérêt. Il me semble donc nécessaire d'avoir de bonnes notions théoriques sur les bases de données pour aborder correctement ce livre. Des administateurs gérant d'autres bases de données trouveront surement un intérêt tout particulier à ce livre.

Retrouvez ce livre sur la rubrique SQL Server : Lien41

Les derniers tutoriels et articles

Quelques conseils pour bien choisir votre boitier

Vous voulez vous monter un nouvel ordinateur, mais vous ne savez pas comment choisir votre boitier ? Alors cet article est fait pour vous, il décrit les différentes choses auxquelles il faut faire attention lors de l'achat d'un boitier.

1. Introduction

Le boîtier va renfermer tous les composants de votre ordinateur, la carte mère, les disques durs, la carte graphique, les lecteurs, … Il y a plusieurs types de boitier et plusieurs normes différentes. De plus, le choix d'un boîtier dépendra aussi de ce que vous allez mettre dedans, une configuration pour la bureautique n'aura pas besoin du même boitier que celle d'un grand joueur. Il faut donc faire attention à toutes ses données avant d'acheter un nouveau boitier.

Nous allons donc voir les points auxquels il faut prêter attention en choisissant un boîtier.

2. Normes

Il existe différentes normes pour les boîtiers : ATX, BTX, Micro-ATX, Flex-ATX, Mini-ITX, SSI MEB et j'en passe. La norme la plus utilisée est la norme ATX et suffira pour la plupart des configurations. Les autres normes telles que Micro-ATX ou Mini-ITX sont destinés à des boitiers de très petite taille. Donc, à moins de vouloir un mini PC pour lequel vous utiliserez une norme telle que le mini-ITX voire pico-ITX ou une carte quadri-processeurs pour laquelle vous aurez du SSI MEB, c'est la norme ATX que vous utiliserez.

En plus d'ATX, on peut aussi trouver la norme BTX qui était censé la remplacer, mais cette norme n'a pas réussi à la supplanter et est de moins en moins utilisée. Vous pourrez néanmoins trouver des boitiers ATX pouvant évoluer vers du BTX. Dans ce cas, les constructeurs proposent un kit de mise à jour qui permet de rendre compatible le boitier pour une configuration BTX. Ce genre de kit contient en général les éléments suivants : une plaque pour la fixation de la carte mère, un panneau arrière, des adaptateurs pour les câbles de l'alimentation et bien souvent les vis nécessaires au montage.

3. Capacité (évolutivité)

L'informatique évoluant très vite, il faut faire attention à avoir un boitier évolutif, pour ne pas se retrouver bloqué. C'est pourquoi, il faut un boitier qui permette d'ajouter des composants supplémentaires. Concrètement, il faut un boitier avec plus d'emplacements que nécessaire.

3.1. Emplacements pour périphériques

Un boitier à deux types d'emplacements :

- Les emplacements visibles (ou externes) : Ce sont les emplacements situés directement sur la face avant du boitier. On y branche surtout les lecteurs CD ou DVD, les lecteurs disquettes ou les lecteurs de carte mémoire. Les personnes ayant une grosse configuration y brancheront aussi peut-être un rhéobus (système de contrôle de ventilateurs) voire même un réservoir de liquide pour leur système à refroidissement liquide.
- Les emplacements invisibles (ou internes) : Ces emplacements sont utilisés le plus souvent pour brancher des disques durs. Certains boitiers intègrent directement des espaces ventilés pour les disques durs. Ils se présentent souvent sous la forme d'un mini-rack qui prend la place de 3 emplacements 5 ¼ muni d'un ventilateur à l'avant.

Ces emplacements ont deux tailles différentes :

- 5 ¼ pouces : Ces emplacements sont surtout utilisés pour les différents types de lecteur et pour les périphériques supplémentaires (rhéobus par exemple, afficheur digital, …). On trouve des kits qui permettent de brancher des périphériques 3 ½ dans de tels emplacements, ce qui peut se révéler très pratique.
- 3 ½ pouces : Essentiellement internes, ces emplacements servent surtout à installer des disques durs. Il y en a aussi souvent un en façade pour le lecteur disquette, mais ceci devient plus rare avec les nouvelles générations de boitiers.

Il est donc très important de choisir un boitier tout d'abord qui puisse renfermer tous vos composants, mais aussi dans lequel il reste de la place pour ne pas devoir changer de boitier trop rapidement. Pour les PC contenant beaucoup de composants, on veillera aussi au refroidissement du boitier (voir prochain chapitre pour plus d'informations).

3.2. Refroidissement

C'est sur le boitier que seront les principaux systèmes de refroidissement. Certains composants ont leurs propres refroidisseurs, mais ils ne font que dissiper la chaleur du composant qui s'accumule alors dans le boitier. Il est donc essentiel de refroidir ce dernier. Il existe plusieurs moyens de refroidir un ordinateur :

- **Ventilation** : C'est le système le plus courant. On pose des ventilateurs en aspiration pour ajouter de l'air frais dans le boitier et des ventilateurs en extraction pour faire

sortir l'air chaud. Il faut donc choisir un boitier qui possède assez d'emplacement pour ventilateurs pour concevoir une bonne circulation de l'air. Il faut aussi faire attention à utiliser des ventilateurs silencieux pour ne pas être trop dérangé par le bruit. N'hésitez pas à choisir un boitier avec des ventilateurs de grande taille (120mm voire 250mm sur les faces latérales), bien que plus gros, ces ventilateurs ne font pas plus de bruit et brassent plus d'air.

- **Watercooling** : Ce système est en général destiné soit aux personnes utilisant de très grosses configurations, soit aux personnes désirant tuner leur ordinateur. On utilise un circuit d'eau pour refroidir les différents composants informatiques de l'ordinateur. L'eau est ensuite refroidie à la fin du circuit par des radiateurs et/ou des ventilateurs puis retourne dans le circuit et ainsi de suite. On trouve quelques boitiers proposant ce genre de système, mais en général, on achète le système en plus du boitier et on le monte soi-même. Si vous voulez le monter, essayez de choisir un PC avec les emplacements pour les tubes déjà prédécoupés, cela vous évitera de devoir couper votre boitier. Il refroidit très efficacement votre boitier, mais il comporte plus de risques, car manipuler de l'eau au milieu de composants informatiques n'est pas une sinécure ; la moindre fuite risquant de faire griller un ou plusieurs de vos composants.
- **Passif** : Il existe certains boitiers entièrement passifs, c'est-à-dire qu'ils n'ont aucun ventilateur et ne font donc aucun bruit. Pour cela, ils utilisent des dissipateurs thermiques et mêmes les parois du boitier sont utilisés comme dissipateur. Ces boitiers sont donc réservés aux personnes ne voulant pas de bruit, mais le problème est qu'ils sont rares et très couteux, sans compter qu'ils sont en général assez imposants et inesthétiques.

D'autres systèmes sont aussi parfois employés, mais souvent en test pour des records de performances, comme par exemple le refroidissement à azote liquide, mais ceci n'est pas conseillé à tout un chacun et peut grandement abimer vos composants. Préférez plutôt une des solutions énoncées plus haut qu'une autre technique plus exotique.

Une chose essentielle est aussi la position du boitier. Il ne sert en effet à rien d'avoir un ordinateur bien ventilé si c'est pour l'enfermer dans une armoire close. Il faut de l'air à votre boitier, sinon c'est la surchauffe assurée.

Pour un PC normal, je ne peux que vous conseiller de choisir un système ventilé. Il faut juste choisir un boitier qui possède assez d'emplacements pour ventilateur pour avoir une bonne circulation de l'air. Néanmoins, si vous voulez tentez le coup du watercooling, sachez que vous pourrez trouver de très bons guides sur internet pour vous aider à vous lancer.

4. Types

En plus des différentes normes énoncées, il existe différents types de boitiers qui influent sur sa forme et sa taille. Voici les différents types que vous pourrez trouver :

- **Mini tour** : Ces boitiers sont destinés aux personnes ne voulant pas utiliser trop d'espace dans leur pièce pour leur PC et ayant une petite configuration. Le problème de ces boitiers est qu'ils n'évacuent souvent pas de manière optimale la chaleur, ils ne sont donc pas destinés à une grosse configuration ni à une utilisation trop intensive. Ils sont en général de norme Micro-ATX.
- **Moyen tour** : Ces boitiers sont les plus communs. Ils sont destinés à une utilisation courante et sont assez évolutifs. On a souvent entre 4 et 5 emplacements externes et plusieurs emplacements internes. La norme la plus utilisée est l'ATX.
- **Grand Tour** : Ces boitiers sont les plus grands et les plus évolutifs. De norme ATX, ils sont utilisés pour les grosses configurations de joueurs ou alors pour un ordinateur avec de nombreux disques durs. Ils possèdent jusqu'à une vingtaine d'emplacements pour périphériques. Ce sont aussi les boitiers qui permettent la meilleure circulation d'air, mais ils prennent bien entendu beaucoup de place et peuvent peser très lourd.
- **Desktop** : Ces boitiers sont destinés à être mis en position horizontale et on posera souvent un écran par dessus. Ils prennent ainsi moins de place et sont plus accessibles. Ils sont souvent à la norme ATX ou Micro-ATX. Ils ont en général 2 ou 3 emplacements externes et 2 ou 3 emplacements internes.
- **Barebone** : Les boitiers de type barebone sont les plus petits boitiers existants à ce jour, ils n'intègrent en général qu'un ou 2 emplacements externes et un interne. La plupart des barebone sont à la norme Micro-ATX.
- **Rack** : Ce type de boitier est destiné à une utilisation professionnelle, on le range dans des armoires de serveurs. On peut mettre plusieurs racks par armoire. La taille d'un rack s'exprime en U (1U = 4.45cm).

5. Matériaux

Un boitier peut être construit avec différents matériaux. Voici les matériaux utilisés pour les boitiers :

- **Acier** : C'est le moins couteux des matériaux, néanmoins c'est aussi le plus lourd et il dissipe assez mal la chaleur. Par contre, il est solide et empêche les vibrations. Si vous optez pour un PC en acier, faites attention à l'épaisseur des parois, il vous faudra des parois épaisses pour limiter les vibrations. Il vous faudra aussi penser à bien refroidir un ordinateur en acier.
- **Aluminium** : Plus couteux que l'acier, mais plus léger et dissipant mieux la chaleur, l'aluminium est néanmoins sujet à plus de vibrations. La plupart des boitiers haut de gamme sont conçus en aluminium.
- **Plexiglas** : les boitiers construits entièrement en plexiglas sont assez rares et très couteux. Néanmoins, cela peut donner quelque chose de très esthétique si votre principale préoccupation est le design de votre ordinateur.

Si vous ne comptez pas déplacer souvent votre boitier, je vous conseille d'utiliser un boitier en acier qui va vous revenir moins cher. Dans le cas contraire, prenez en boitier en aluminium qui sera beaucoup plus léger.

6. Design

Bien qu'inutile pour certains, le design d'un boitier va se révéler très important pour d'autres. Certains boitiers intègrent des vitres en plexiglas qui permettent de voir l'intérieur du PC. Certains ont tout simplement une forme spéciale, par exemple avec une porte à l'avant. D'autres encore intègrent des éléments lumineux à l'intérieur du PC.

En ce qui me concerne, pour le tuning du PC, si vous n'avez pas envie de faire quelque chose vous-même, le mieux est d'acheter un PC déjà moddé. Mais si vous avez envie de prendre le temps de travailler sur votre PC, vous pourrez faire quelque chose de très esthétique; je vous conseille alors d'acheter un boitier assez

simple afin que vous puissiez facilement le modifier et acheter les différents composants pour le tuning à coté.

6. Autres

On peut encore faire attention à certaines choses lors de l'achat d'un boitier.

Pour commencer les ports en façade. Presque tous les boitiers intègrent maintenant des ports en façade, souvent ce sont 2 ports USB, un port pour un casque audio, un port micro et parfois un port Firewire. Ces ports sont très pratiques car très accessibles. Veillez donc toujours à avoir au moins des ports USB en façade, on les utilise très souvent pour brancher une clef USB.

Ensuite, vous pouvez aussi choisir un boitier qui possède des filtres à air en façade. Ces filtres vont beaucoup réduire la poussière à l'intérieur du boitier, ce qui va permettre une meilleure circulation et moins de chaleur et accessoirement vous éviter de devoir nettoyer votre boitier toutes les 2 semaines.

8. Conclusion

Pour conclure, je dirais que la première chose à laquelle il faut faire attention dans le choix d'un boitier est son évolutivité pour ne pas être bloqué peu de temps après l'achat du PC. Ensuite, il faut bien sûr aussi faire attention à ce qu'il soit bien refroidi et qu'il ne fasse pas trop de bruit. Enfin, vous pouvez prendre en compte le design des boitiers.

Une fois l'achat fait, il ne faudra pas non plus oublier de nettoyer le boitier et son contenu au grand minimum une fois par an, mais une fois tous les 3 mois serait plus avisé.

Je vous invite aussi à consulter cet article (Lien42) qui vous propose différentes configurations adaptées à différents besoins.

Retrouvez l'article complet de Baptiste Wicht en ligne : Lien43

Activer le SSL dans IIS 7.0 sous Windows Vista

Dans cet article nous allons voir comment utiliser SSL dans IIS 7.0 sans pour autant devoir se procurer un certificat en bonne et due forme auprès d'une autorité de certification reconnue, ce qui est relativement onéreux et pas forcément indispensable quand il s'agît comme nous de l'utiliser sur une machine de développement et non dans un environnement de production. Il y a de cela presque trois ans j'avais écris un article sur le même sujet mais à l'époque sous IIS 5.0 qui était livré avec Windows 2000. Voyons depuis le chemin parcouru, et apprécions sans plus tarder la simplicité qu'apporte le nouveau IIS en la matière.

1. Introduction

Windows Vista inclut une nouvelle version de Internet Information Services, il s'agît de la 7.0, qui succède côté système d'exploitation client à la version 5.1 intégrée à Windows XP Pro. La version 6.0 était quand à elle intégrée à Windows Server 2003. Le IIS nouveau cru possède de nombreuses améliorations, qui, pour certaines sont très intéressantes. Je ne m'attarderai cependant pas sur celles-ci car ce n'est pas l'objet de cet article, mais si vous souhaitez en savoir plus je vous renvoie vers ce très bon site en anglais (Lien44).

Dans cet article nous allons voir comment utiliser SSL dans IIS 7.0 sans pour autant devoir se procurer un certificat en bonne et due forme auprès d'une autorité de certification reconnue, ce qui est relativement onéreux et pas forcément indispensable quand il s'agît comme nous de l'utiliser sur une machine de développement et non dans un environnement de production. Il y a de cela presque trois ans j'avais écris un article sur le même sujet mais à l'époque sous IIS 5.0 qui était livré avec Windows 2000. Voyons depuis le chemin parcouru, et apprécions sans plus tarder la simplicité qu'apporte le nouveau IIS en la matière.

2. Qu'est-ce que SSL ?

SSL, pour Secure Socket Layer, est un protocole de sécurité permettant d'encrypter les données échangées entre serveurs Web et clients. Ce protocole est extrêmement répandu, par exemple la quasi-totalité des sites d'e-commerce l'utilise. La mise en place de SSL sur IIS 7.0 nécessite l'emploi d'un certificat de sécurité numérique qui va servir à encoder et décoder les données chiffrées transitant dans un flux SSL.

3. Installation d'IIS 7.0

Remarque : quelque soit le processus que vous utilisez vous devez avoir à l'esprit que vous devez être titulaire des privilèges administrateur sur la machine sur laquelle vous souhaitez installer IIS 7.0. Pour cela en fonction du compte que vous utilisez et du paramétrage de l'UAC de Windows Vista vous devrez fournir les informations (identifiant et mot de passe) d'un compte administrateur, ou autoriser l'élévation du niveau de privilège au niveau administrateur.

Deux possibilités s'offrent ici à nous. La ligne de commande ou l'interface graphique. Voyons comment procéder, et les avantages et inconvénients des deux solutions.

La ligne de commande : Windows Vista intègre un nouvelle utilitaire nommé Package Manager qui permet, entre-autres, d'ajouter ou de supprimer des fonctionnalités de Windows. Grâce à cet utilitaire utilisable depuis la ligne de commande vous allez pouvoir choisir chaque fonctionnalité à installer, sans pour autant devoir dérouler les multiples sous menus de l'installation de IIS depuis l'assistant habituel de gestion des fonctionnalités Windows. De plus, cette solution s'avère idéale pour les administrateurs qui veulent déployer IIS 7.0 sur de nombreuses machines en utilisant un simple script. Bref, je ne vais pas vous faire la promotion de la ligne de commande, mais sachez qu'ici c'est l'option que je préfère.

Voici la commande permettant d'installer l'ensemble des fonctionnalités d'IIS 7.0 :

```
start /w pkgmgr /iu:IIS-WebServerRole;IIS-
WebServer;IIS-CommonHttpFeatures;
IIS-StaticContent;IIS-DefaultDocument;IIS-
DirectoryBrowsing;IIS-HttpErrors;
IIS-HttpRedirect;IIS-ApplicationDevelopment;IIS-
ASPNET;IIS-NetFxExtensibility;
IIS-ASP;IIS-CGI;IIS-ISAPIExtensions;IIS-
ISAPIFilter;IIS-ServerSideIncludes;
IIS-HealthAndDiagnostics;IIS-HttpLogging;IIS-
LoggingLibraries;IIS-RequestMonitor;
IIS-HttpTracing;IIS-CustomLogging;IIS-ODBCLogging;IIS-
Security;IIS-BasicAuthentication;
IIS-WindowsAuthentication;IIS-DigestAuthentication;IIS-
ClientCertificateMappingAuthentication;
IIS-IISCertificateMappingAuthentication;IIS-
URLAuthorization;IIS-RequestFiltering;
IIS-IPSecurity;IIS-Performance;IIS-
```

```
HttpCompressionStatic;IIS-HttpCompressionDynamic;
IIS-WebServerManagementTools;IIS-ManagementConsole;IIS-
ManagementScriptingTools;
IIS-ManagementService;IIS-
IIS6ManagementCompatibility;IIS-Metabase;IIS-
WMICompatibility;
IIS-LegacyScripts;IIS-LegacySnapIn;IIS-
FTPPublishingService;IIS-FTPServer;IIS-FTPManagement;
WAS-WindowsActivationService;WAS-ProcessModel;WAS-
NetFxEnvironment;WAS-ConfigurationAPI
```

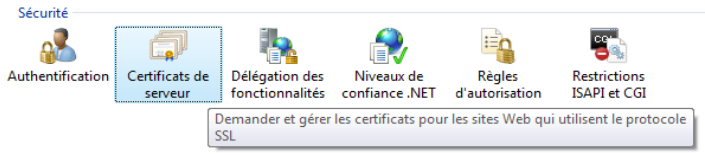
Assistant de gestion des fonctionnalités Windows : comme pour les versions précédentes de Windows, celui-ci est accessible depuis le panneau de configuration. Ensuite il faut double cliquer sur "Programmes et fonctionnalités". Cliquez sur "Activer ou désactiver des fonctionnalités Windows". Dans la liste déroulante sélectionnez toutes les sous fonctionnalités de "Services Internet (IIS)" que vous souhaitez installer. Après avoir coché les fonctionnalités souhaitées, il ne vous reste plus qu'à cliquer sur "Ok", l'assistant va alors configurer Windows et IIS de manière adéquate.

A ce stade votre IIS est installé et fonctionnel, voyons maintenant comment générer notre certificat qui nous permettra d'utiliser SSL pour sécuriser tout ou partie d'une application web.

4. Génération d'un certificat

Contrairement à ce que l'on rencontrait dans les versions précédentes de IIS, 5.0 et 6.0, ici tout est fait pour rendre l'opération rapide et simplissime. Tout d'abord, allez dans le panneau de configuration, puis dans les outils d'administration. Double-cliquez sur "Gestionnaire des services Internet (IIS)" pour lancer la toute nouvelle console de gestion d'IIS 7.0. Cette opération nécessite des privilèges administrateur.

Dans la colonne de gauche, présentant l'arborescence de votre serveur IIS, placez vous sur l'élément racine, puis double cliquez sur "Serveur de certificats" dans la partie centrale de la console.



Dans la partie droite de la console, dans la colonne "Action", cliquez sur "Créer un certificat auto-signé". Un assistant se lance et vous demande de saisir un nom convivial pour le certificat à générer. Cliquez sur "Ok". Votre certificat est désormais créé, vous voyez à quel point cette opération est d'une grande complexité...

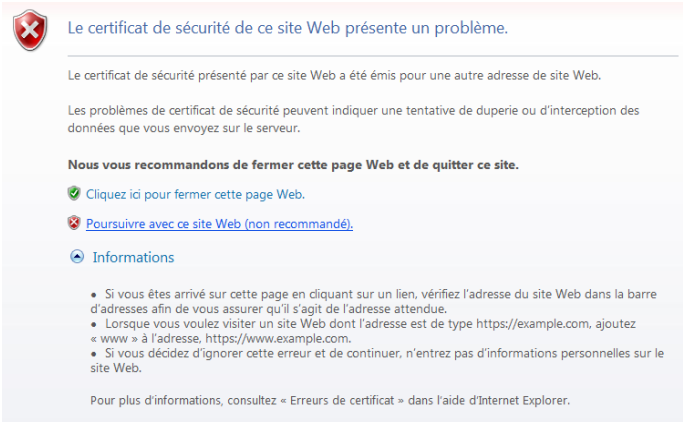
Vous devez maintenant voir apparaitre ce nouveau certificat dans la partie centrale de la console. Au passage, remarquez que ce certificat à une validité d'un an. En double-cliquant sur le certificat vous affichez une fenêtre présentant les différentes caractéristiques de celui-ci. Reste maintenant à utiliser ce certificat pour sécuriser les échanges à l'aide de SSL.

5. Mise en place du SSL

Nous allons devoir indiquer à IIS que nous souhaitons utiliser ce certificat pour sécuriser le flux SSL sur notre serveur. Cela s'effectue également depuis la console de gestion d'IIS. Tout d'abord, il faut sélectionner le site web sur lequel nous souhaitons activer le SSL, puis cliquer dans la colonne de droite sur "Liaisons" ("Bindings" pour les versions anglaises) puis sur le bouton "Ajouter". Nous allons ajouter un nouveau type de liaison. Sélectionnez https dans la liste déroulante, laissez le port 443 inchangé, il s'agit du port de communication par défaut de https, le port par défaut pour http est je vous le rappelle le port 80. Sélectionnez ensuite le certificat que nous venons de créer dans la liste prévue à cet effet, cliquez sur "OK", et voilà c'est terminé.

Pour vérifier que votre site est bien accessible par le biais du protocole SSL tapez dans votre navigateur préféré l'URL de votre site, bien évidemment précédé de https et non de http. Un petit cadenas doit apparaitre sur votre navigateur signalant que la connexion est sécurisée.

Remarque : si vous utilisez IE 7.0 il est possible que votre navigateur affiche un message d'erreur vous informant que ce certificat n'est pas valide, c'est tout à fait normal. En effet ce certificat n'a pas été crée par une autorité de certification reconnue. Mais ceci n'est pas grave car nous utilisons ce certificat à des fins de test et non en production, vous pouvez donc ignorer l'avertissement et consulter tranquillement votre site en SSL en cliquant sur "Poursuivre avec ce site Web (non recommandé)".



6. Conclusion

Comme vous avez pu le voir c'est absolument simplissime, depuis la version 5.0 le processus à été grandement simplifié, notamment au niveau de la génération du certificat, en effet IIS effectue desormais cette tâche de manière autonome. Vous devez savoir qu'il est possible de paramétrer très finement quelle partie de votre application doit être accédée en SSL ou non, la console de IIS vous autorise de telles configurations en accédant aux paramètres SSL de l'objet sélectionné, cela peut être par exemple un site web ou un répertoire virtuel. J'espère vous avoir montré à quel point cette opération est simple et que vous n'hésiterez plus a tester vos applications en cours de développement avec SSL lorsque cela est nécessaire. Il est évident que vu la nature de notre certificat il ne peut servir qu'en environnement de développement, mais le processus en production avec un certificat émis par une autorité de certification dite de confiance serait à peu de chose près le même.

Retrouvez l'article de Ronald Vasseur en ligne : Lien45

Installation de Subversion sous Debian

Cet article présente l'installation d'un serveur Subversion sur une distribution Linux Debian ainsi que les bases de gestion d'un projet sur ce même serveur.

Il présentera ensuite une application Web, USVN, qui propose une interface simple de gestion des dépôt Subversion.

1. Qu'est-ce que subversion ?

1.1. Qu'est-ce qu'un VCS ?

Subversion est un VCS, Version Control System, ou logiciel de gestion de version. C'est un logiciel qui permet à plusieurs développeurs de travailler sur un même projet, voir sur un même fichier simultanément.
Il est responsable de la gestion des accès concurrents, ou plus exactement des modifications concurrentes sur un même fichier. De plus si les modifications ne sont pas conflictuelles, Subversion assure la fusion de celles-ci afin d'avoir au final une version du projet qui comprend toutes les modifications.

Dans les cas où les modifications portent sur les même lignes, Subversion aura besoin d'une intervention d'un utilisateur afin de résoudre manuellement ces conflits.

1.2. Pourquoi Subversion ?

Subversion a été développé dans le but de remplacer à terme CVS en comblant quelques lacunes inérantes à ce dernier :

- Atomicité
- Gestion du renommage / déplacement
- Gestion des révisions
- Gestion des accès

1.3. Atomicité

Dans CVS les modifications envoyées par un développpeur étaient intégrées dans le dépôt une par une dès leur réception. Si pour une raison ou l'autre, l'envoi de ces modifications était intérrompu (problème réseau, plantage du système d'exploitation du coté client, ...), les modifications déjà reçues étaient été intégrées.
Dans ce cas précis, le dépôt est dans un état incohérent, et le projet n'est plus forcément compilable.

Afin d'éviter cela, Subversion intégre les modifications uniquement lorsque toutes celles-ci ont été reçues intégralement et qu'aucun problème n'est survenu.
Le cas échéant, aucune modification n'aura été intégrée, et le développeur devra faire une autre tentative, après résolution des problèmes éventuels.

Ce système d'intégration est comparable au système de transactions des bases de données commit/rollback.

1.4. Gestion des déplacements / renommages

Une autre faiblesse de CVS est la gestion, ou plus exactement l'absence de gestion, du renommage/déplacement des fichiers. En effet, CVS perçoit ces actions commme la suppression du fichier source, et la création du fichier destination. Ce qui a pour conséquence que l'historique du fichier original n'est pas lié au nouveau fichier.

Dans Subversion, si un fichier est déplacé ou renommé à l'aide de la commande svn move, l'historique reste associé au fichier destination, ce qui permet de voir toutes ses modifications depuis le début.

Attention, si le déplacement n'est pas fait avec la commande svn, l'historique sera perdu de la même manière que dans CVS.

1.5. Gestion des répertoires

Un autre avantage de Subversion est sa gestion des répertoires. Pour chaque révision, une liste des fichiers contenus dans un répertoire est gardé.
Cela a pour avantage de pouvoir comparer le contenu d'un répertoire par rapport à une ancienne version rapidement.

1.6. Révisions

La notion des révisions entre Subversion et CVS est différente. Dans ce dernier une révision était associée à un fichier en particulier.

Cela peut paraître logique, mais cela pose un problème : Comment récupérer une copie du dépôt à un moment donné ? Un fichier pouvant être à la révision 3, tandis qu'un autre à la révision 10, et un plus récent à la révision 1.

Dans Subversion, la notion de version est appliquée sur l'ensemble du dépôt. Dès qu'une modification est apportée à un fichier dans le dépôt, c'est l'ensemble des données qui passe au numéro de révision suivant.

Il est dès lors plus simple de retrouver l'état d'un dépôt à un moment donnée, en gardant une trace de la révision à ce moment là.

1.7. Gestion des accès

Comme il sera démontré plus loin dans cet article, la gestion des accès peut être très fine, spécialement lorsque Subversion est lié à Apache.
Il est dès lors possible de spécifier différents accès par utilisateur, par groupe, et ce, pour chaque fichier/répertoire dans le dépôt, et non pas un accès global à celui-ci.

2. Subversion sous debian

2.1. Installation

L'installation de Subversion se fait simplement par la commande :

```
aptitude install subversion
```

Cette commande installe le client Subversion, différentes commandes de gestion de dépôt ainsi qu'un exécutable serveur.
Sur une distribution Debian Etch, la version est 1.4.2.

2.2. Gestion d'un dépôt

Cela fait, il est nécessaire de créer un premier dépôt.
Dans le cadre de cet article, le répertoire de base des dépôts sera /var/subversion/depot/

La création d'un dépôt pour un projet nomdemonprojet se fait via la commande :

```
svnadmin create /var/subversion/depot/nomdemonprojet
```

Le résultat devrait être un répertoire nomdemonprojet crée dans /var/subversion/depot/

2.3. Utilisation du client

Dès lors, il est déjà possible de travailler sur celui-ci localement (sur la même machine que le dépôt lui même). L'accès à distance nécessitant un peu de configuration, cela sera expliqué par après.

En attendant, voici un petit descriptif des commandes de bases du client Subversion.

2.3.1. Check out

La première étape dans l'utilisation de Subversion est de récupérer une copie locale du dépôt :

```
svn co file:///var/subversion/depot/nomdemonprojet
```

Si tout c'est bien déroulé, un répertoire nomdemonprojet a du être créé dans le répertoire courant.

Ce répertoire doit contenir les fichiers de la dernière révision disponible sur le dépôt.
Dans ce cas-ci, le projet venant d'être créé, ce répertoire est vide.

2.3.2. Ajout de fichiers/répertoires

Dans ce répertoire, vous êtes libre de travailler normalement : création de répertoire, ajout de fichier.
Dans le cadre de cet article, la hiérarchie de fichiers/répertoires suivante a été créée :

```
nomdemonprojet
  + src
    + java
      + com
        + developpez
            + hikage
                + TestMain.java
                + Application.java
    + webapp
      + WEB-INF
        + web.xml
```

Il est ensuite nécessaire d'informer Subversion que ces fichiers doivent être pris en compte lors de la prochaine publication des modifications sur le dépôt :

```
svn add src
```

Le résultat de cette commande est :

```
debian:~/nomdemonprojet# svn add src
A         src
A         src/webapp
A         src/webapp/WEB-INF
A         src/webapp/WEB-INF/web.xml
A         src/java
A         src/java/com
A         src/java/com/developpez
A         src/java/com/developpez/hikage
A         src/java/com/developpez/hikage/TestMain.java
A
src/java/com/developpez/hikage/Application.java
```

2.3.3. Intégration des modifications sur le serveur

La dernière commande n'a fait que mettre une "étiquette" sur des fichiers, mais ceux-ci n'ont pas encore été envoyé vers le dépôt.
Cette action est réalisée via la commande :

```
svn commit src -m "Message expliquant le modification
effectuées"
Ajout           src
Ajout           src/java
Ajout           src/java/com
Ajout           src/java/com/developpez
Ajout           src/java/com/developpez/hikage
Ajout
src/java/com/developpez/hikage/Application.java
Ajout
src/java/com/developpez/hikage/TestMain.java
Ajout           src/webapp
Ajout           src/webapp/WEB-INF
Ajout           src/webapp/WEB-INF/web.xml
Transmission des données ...
Révision 1 propagée.
```

Le paramètre **-m "Mon Message"** spécifie un message qui sera associé à la publication (et donc à la nouvelle révision).
Il est courant d'expliquer comme message le but des modifications réalisées : ajout d'une fonctionalité, correction d'un bug, ...

2.3.4. Intégration, le retour

Imaginons maintenant que du code a été ajouté dans le fichier TestMain.java (qui était jusqu'ici un simple fichier vide), et que lors de la publication des modifications, le résultat affiche :

```
debian:~/nomdemonprojet/src/java/com/developpez/hikage#
svn commit TestMain.java -m "Ajout du code par
developpeur 1"
Envoi           TestMain.java
svn: Échec de la propagation (commit), détails :
svn: Le chemin
'/src/java/com/developpez/hikage/TestMain.java' est
obsolète dans la transaction '3-1'
```

Le message informe que le fichier du dépôt est plus récent (une révision supérieure) que celle disponible en locale.
Il est donc nécessaire de mettre à jour cette dernière grâce à la commande :

```
svn update TestMain.java
C    TestMain.java
Actualisé à la révision 3.
```

La mise à jour a été effectuée mais un conflit (le C à coté du nom du fichier) est apparu.
Lors d'un conflit, Subversion crée trois nouveaux fichiers dans le répertoire, ici TestMain.java.mine (qui est une copie de la version modifiée localement), TestMain.java.r1 (qui est la version locale avant modification, c'est à dire celle récuperée lors du check out ou de la dernière mise à jour), et TestMain.java.r3 qui est la dernière version sur le serveur.

Le fichier TestMain.java quand à lui possède des informations sur les différences entre chaque fichiers.
Il est donc nécessaire de réaliser la fusion manuellement, en gardant les bonnes lignes.

Dans le cas de la ligne de commande, la fusion doit être réellement faite à la main.
Lors de l'utilisation de client graphique, il est fréquent d'avoir une fenêtre qui affiche les différences entre les fichiers et propose une interface simple pour réaliser la fusion.

Une fois la fusion effectuée, il faut dire à Subversion que le problème à été résolu :

```
svn resolved TestMain.java
Conflit sur 'TestMain.java' résolu
```

En plus de supprimer l'étiquette 'Conflit' (qui empêche le fichier d'être envoyé sur le dépôt), cette commande supprime les trois fichiers temporaires cités plus haut.

Le cas du "conflit" est le seul qui nécessite l'intervention manuelle. Lors d'une mise à jour de la copie locale (svn update), les informations possibles associées au fichier peuvent être :

- **C** : Conflit, c'est le cas présenté dans cet article
- **A** : Ajout, un nouveau fichier a été ajouté sur le dépôt et est donc récupéré dans la copie locale
- **D** : Supression, un fichier a été marqué comme supprimé, et donc la copie locale est supprimée aussi
- **G** : Fusion, des modifications ont été réalisées sur un fichier qui a été modifié dans la copie locale, mais les modifications ne sont pas conflictuelle.
  Subversion intégre donc ces modifications dans la copie locale
- **M** : Modification, des modifications ont été réalisées sur un fichier qui n'a pas été modifié localement, les modifications sont donc intégrées dans la version locale

Voila qui finit la présentation des commandes de base de Subversion. Il en existe beaucoup d'autre afin de créer des patchs entre version (svn diff), annuler des modifications locales (svn revert), et bien d'autres encore qui nécessiteraient un article à part afin d'être présenté correctement.

2.4. Utilisation en réseau

Jusqu'ici le serveur était accessible uniquement en local et son utilité est donc restreinte.
Pour mettre à disposition les dépôts à des clients distants, il existe différents moyens possèdant chacun leurs avantages et leurs inconvénients.

2.4.1. SVNServe

La première méthode est d'utiliser SVNServe, un démon fourni dans le package Subversion. Celui-ci peut être configuré en serveur Standalone ou via InetD.

L'avantage de cette méthode est qu'il ne nécessite pas d'autre dépendance.

L'inconvénient est qu'il écoute sur un port propre (qui est bien sur configurable), et qu'il risque donc d'être bloqué par des pare-feux.

2.4.2. SSH

Il est aussi possible d'accèder au dépôt via un tunnel SSH. Cela nécessite bien sûr un serveur SSH installé sur la machine qui héberge le dépôt, mais c'est souvent le cas sur des machines Linux.

Du côté du client, il est nécessaire que celui-ci gère ce type d'accès. Parmi ceux-ci, citons Tortoise SVN, SmartSVN ou encore Subclipse

2.4.3. Module apache

La dernière méthode, et la plus intéressante, est d'utiliser un module Apache pour Subversion.

Le principal intérêt par rapport à l'accès via SvnServe ou SSH est que le port HTTP est très rarement filtré par les pare-feux, et donc est accessible de partout.
De plus l'authentification étant effectuée via les modules d'Apache, les moyens d'authentification de celui-ci (fichier d'utilisateur, annuaire LDAP, base de données) sont disponibles.

Comme il a été dit au début de cet article, l'intégration Apache / Subversion fourni un dispositif de gestion des droits d'accès plus fine.

Dans le cadre de ce tutoriel, c'est cette méthode qui sera expliquée.

2.4.3.1. Installation

La première chose à faire est d'installer Apache si ce n'est pas encore le cas :

```
aptitude install apache2
```

Ensuite, il faut installer le module svn en lui même.
Sous debian Sarge :

```
aptitude install libapache2-mod-dav libapache2-svn
```

Tandis que sous debian Etch, cela se fait via :

```
aptitude install libapache2-mod-svn
```

2.4.3.2. Configuration

Une fois le module installé, il faut configurer Apache afin d'intégrer le dépôt.
Pour ce faire, nous allons configurer Apache afin que l'URL de type http://monserveur/svn fournisse un accès au dépôt.

Tout d'abord, il faut informer Apache qu'il doit prendre en compte le module SVN :

```
a2enmod dav_svn
```

Ensuite, le fichier de configuration /etc/apache2/mod-available/dav_svn.conf doit être édité :

```
<Location /svn>
        DAV svn
        Require valid-user
        SVNParentPath /var/subversion/depot/
        AuthType Basic
        AuthName "Mon dépôt"
        AuthUserFile /var/subversion/conf/htpasswd
        AuthzSVNAccessFile /var/subversion/conf/access
</Location>
```

Ligne	Description
Location /svn	Spécifie que le dépôt sera accessible via http://ip/svn Ou ip représente l'adresse ip du serveur
DAV svn	Active la gestion SVN sur ce répertoire
Require valid-user	Nécessite un utilisateur valide afin d'accèder à ce répertoire
SVNParentPath	Configure le chemin vers le répertoire parent des dépôts
AuthType	Spécifie le type d'authentification
AuthName	Spécifie le nom qui sera affiché dans la boite de demande de mot de passe
AuthUserFile	Spécifie le fichier des utilisateurs
AuthzSVNAccessFile	Spécifie le fichier de gestion des accès

2.4.3.3. Gestion des utilisateurs

Cela fait, l'étape suivante est de créer le fichier /var/subversion/conf/htpasswd qui contiendra les utilisateurs et leurs mots de passe.
La création du premier utilisateur est réalisée via la commande :

```
htpasswd -c /var/subversion/conf/htpasswd hikage
```

hikage représente le nom de l'utilisateur, et son mot de passe sera demandé par la commande.
Le paramètre -c spécifie qu'il est nécessaire de crée le fichier. L'ajout des prochains utilisateurs se fera donc via la commande :

```
htpasswd /var/subversion/conf/htpasswd utilisateur2
```

Une fois les utilisateurs créés, il reste encore à configurer leurs accès. Cela est réalisée dans le fichier /var/subversion/conf/access .
Le contenu de ce fichier dans le cadre de cet article pourrait être :

```
[groups]
developpez  = hikage, utilisateur2

[nomdemonprojet:/]
@developpez = rw
* = r

[projetprivehikage:/]
hikage = rw
* = r

[projetprivehikage:/documentation/utilisateur]
auteurdoc = rw
```

La section [groups], comme son nom le laisse sous entendre, permet de définir des groupes. Ici un seul groupe (**developpez**), dont les membres sont hikage et utilisateur2, est créé

La deuxième section ([nomdemonprojet:/]) définit les accès globaux au projet **nomdeprojet**. Ici le groupe développezcom (@developpezcom) possède un droit en lecture/écriture, tandis que tout les autres (représenté par *) ont un accès en lecture seule.

La troisième section définit les accès globaux à un deuxième projet (projetprivehikage), accessible en écriture uniquement par hikage mais lisible par tous.

Remarquez que le fichier de configuration des droits d'accès peut être commun à plusieurs projets.

La dernière section redéfinit les droits d'accès au répertoire documentation/utilisateur du projet projetprivehikage. Celui-ci sera accessible en écriture par auteurdoc.

2.4.3.4. Relancement du serveur

Ces étapes réalisées, il est nécessaire de relancer Apache afin qu'il prenne ces changements en compte :

```
/etc/init.d/apache2 restart
```

Retrouvez l'article de Gildas Cuisinier en ligne avec l'instlalation d'une interface web de gestion SVN : Lien46

Les blogs Linux

Sortie de la GPL v3

Hier, 29 juin 2007, sont sorties les nouvelles versions des licences GPL et LGPL, à savoir la très attendue version 3.0

Cette nouvelle version était très attendue de la communauté libre, car elle doit implicitement empêcher des phénomènes comme la tivoization (Lien47).

plus d'infos : Lien48

Retrouvez ce billet sur le blog de Gorgonite : Lien49

Les derniers tutoriels et articles

Tutoriel XGQL

Dans un précédent article (Lien50), nous avions réalisé une brève introduction à l'API XGQL autour d'une étude de cas. Ici, nous allons plus nous intéresser au formalisme du langage et aux nouveautés de la version 2.0

1. Introduction

XGQL est un langage procédural utilisant le formalisme XML permettant d'agréger de nombreuses sources de données :

- Plusieurs bases de données (depuis la version 2.0)
- Flux XML, SOAP, Texte
- Session, request et cookie en mode web (depuis la version 2.0)

Via les protocoles suivants :

- File system (Local ou points de montage)
- HTTP
- FTP
- TCP
- JDBC

Nous allons voir comment utiliser ce formalisme.

2. Au commencement fut le namespace

Un script XGQL est déclaré dans le namespace <xgql:myTag/>. C'est pour cela qu'un listing XGQL devra débuter de la façon suivante :

```
<xgql:root xmlns:xgql="http://www.symeria.com/xgql/">
</xgql:root>
```

3. Les variables

Une variable XGQL se déclare de la façon suivante :

```
<xgql:var name="maVariable">Ma variable</xgql:var>
```

Par défaut, une variable n'a pas de type, elle peut être de type :

- Entier
- Chaîne de caractères
- XML

Contrairement au XSLT, il est possible de modifier le contenu d'une variable :

```
<xgql:var name="maVariable">$maVariable et mon nouveau
contenu</xgql:var>
```

Attention : Si vous déclarez une variable dès le début du traitement, celle-ci sera globale. Par contre, si vous la déclarez dans une condition, une fonction ou une boucle, elle sera locale.

3.1. Opérateurs arithmétiques

Pour travailler sur les variables, XGQL permet d'utiliser les opérateurs mathématiques de base.

Opérateur	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division
Mod	Modulo

3.2. Affectation de variable

L'affectation de variable s'effectue de la même façon que la déclaration de celle-ci :

```
<xgql:var name="maVariable">1</xgql:var>
```

A ce moment, maVariable vaut 1.
Il est possible d'utiliser tous types d'opérateurs mathématiques :

```
<xgql:var name="maVariable"><xgql:process>$maVariable +
1</xgql:process></xgql:var>
<!--maVariable est à 2 -->
<xgql:var name="maVariable"><xgql:process>$maVariable *
$maVariable </xgql:process></xgql:var>
<!--maVariable est à 4 -->
<xgql:var name="maVariable"><xgql:process>$maVariable
mod $maVariable </xgql:process></xgql:var>
<!--maVariable est à 0 -->
```

L'instruction **xgql:process** indique qu'il s'agit d'une opération. Sinon, il s'agirait d'une concaténation de chaînes.
xgql:process : permet aussi de réaliser des opérations XQuery, comme par exemple :

```
<xgql:var name=' maVariable'>
<xgql:process>
let $doc:=document{<account
id='42'><name>Aurel</name><firstnm>Marc</firstnm><log>m
artin_p</log><pwd>mypassword</pwd></account>}
return $doc/account/data(@id)
</xgql:process>
</xgql:var>
```

Ici, maVariable sera à 42, puisqu'il s'agit de la valeur de l'attribut id de l'élément account.

3.3. Opérateurs de chaînes

Le but premier du script XGQL étant de générer du XML, il est évidemment possible de travailler sur les chaînes.

```
<xgql:var name="maVariable">Ma variable</xgql:var>
```

maVariable vaut 'Ma variable'.

```
<xgql:var name="maVariable">$maVariable et mon nouveau
contenu</xgql:var>
```

maVariable vaut 'Ma variable et mon nouveau contenu'. Bien entendu, les possibilités de traitement de chaînes sont bien plus poussées, puisqu'il est également possible d'accéder à l'ensemble des fonctionnalités XQuery pour le traitement des chaînes. Exemple de concaténation de chaîne en XQuery :

```
< !-- Renvoi montestmontest -->
<xgql:var name="maVariable"><xgql:process>let $doc :=
string('montest') return concat($doc,
$doc)</xgql:process></xgql:var>
```

Mais nous ne nous étendrons pas plus sur les fonctions XQuery car ces dernières sont déjà décrites dans d'autres tutoriaux.

3.4. Variables externes

xgql:param est la commande qui permet à un script XGQL d'être initialisé par des paramètres extérieurs quand il est utilisé en mode API ou *generator cocoon*.

Exemple de paramètre

```
<xgql:param name="externalParam"/>
```

Cette déclaration est à placer au début du script.

4. Les instructions de contrôle

Les instructions de contrôle constituent le squelette du script XGQL. Ce sont ces instructions qui procurent à XGQL sa capacité décisionnelle permettant de contrôler les flux d'exécution du programme.
Les instructions capables de modifier l'exécution du script en fonction des données métiers sont les suivantes :

- **xgql:if** : exécute un bloc de code lorsque certaines conditions sont satisfaites.
- **xgql:while** : exécute plusieurs fois un même bloc de code.

4.1. Les tests conditionnels

4.1.1. Instruction xgql:if

L'une des fonctionnalités les plus importante est l'instruction **xgql:if**. L'instruction **xgql:if** exécute les instructions si la condition renvoie la valeur booléenne true.
Exemple de déclaration d'un bloc conditionnel :

```
<xgql:if test="1 = 1">
	instruction
</xgql:if>
```

Ces instructions peuvent être imbriquées :

```
<xgql:var name="condition1">true</xgql:var>
<xgql:var name="condition2">2</xgql:var>
<xgql:if test="'$condition1' = 'true'">
<xgql:if test="'$condition2' = '2'">
		instruction
</xgql:if>
</xgql:if>
```

4.1.2. La branche conditionnel

Pour simplifier l'écriture des scripts, il est possible d'exécuter un bloc d'instructions spécifiques si le résultat de la condition est à la valeur booléenne false.
Avec le script suivant :

```
<xgql:root xmlns:xgql="http://www.symeria.com/xgql/">
<xgql:var name="condition1">true</xgql:var>
<xgql:var name="condition2">2</xgql:var>
<result>
	<xgql:if test="'$condition1' = 'true'">
		<xgql:if test="'$condition2' = '3'">
			<test1/>
			<xgql:else>
				<test2/>
			</xgql:else>
			<test3/>
		</xgql:if>
		<xgql:else>
				<test4/>
		</xgql:else>
		<test5/>
	</xgql:if>
</result>
</xgql:root>
```

on obtient le résultat suivant :

```
<result>
	<test2></test2>
	<test5></test5>
</result>
```

Dans le deuxième saut conditionnel, il est possible de mettre des instructions après le **xgql:else**, par contre, il n'est pas possible de mettre plusieurs branches conditionnelles dans le même **xgql:if**.

Un exemple qui ne fonctionnera pas :

```
<xgql:root xmlns:xgql="http://www.symeria.com/xgql/">
<xgql:var name="condition1">true</xgql:var>
<xgql:var name="condition2">2</xgql:var>
<result>
	<xgql:if test="'$condition1' = 'true'">
		<xgql:if test="'$condition2' = '3'">
			<test1/>
			<xgql:else>
				<test2/>
			</xgql:else>
			<test3/>
			<xgql:else>
				<test2/>
			</xgql:else>
		</xgql:if>
		<xgql:else>
				<test4/>
		</xgql:else>
		<test5/>
	</xgql:if>
</result>
</xgql:root>
```

4.1.3. Opérateurs de comparaisons

Les opérateurs de comparaisons servent à comparer deux valeurs (variable et constante). Ils renvoient toujours un booléen.

Opérateur	Description
= ou eq	Egal à

!=	Différent de
lt ou <	Inférieur à
gt ou >	Supérieur à
<=	Inférieur ou égal à
>=	Supérieur ou égal à

4.1.4. Opérateurs logiques

Les opérateurs logiques peuvent être combinés entre eux pour obtenir des résultats booléens plus fins.

Opérateur	Description
and	Et logique
or	Ou logique
not()	Négation

4.2. Les boucles

La boucle **xgql:while** est une expression proche du **xgql:if**. Il évalue une expression booléenne et exécute un bloc de code aussi longtemps que l'expression booléenne renvoie true. Il n'existe pas de branchement conditionnel sur le **xgql:while**.

```
<xgql:var name="condition1">0</xgql:var>
  <xgql:while test='$condition1 lt 5'>
              $condition1
              <xgql:var
name="condition1"><xgql:process>$condition1 +
1</xgql:process></xgql:var>
    </xgql:while>
```

Cet exemple renvoie le résultat suivant : 1234

5. Les procédures

Une procédure est un bloc de script qui peut être invoqué à tout moment. Une procédure accepte un ou plusieurs paramètres, mais ne peut pas renvoyer de valeur.
Une procédure se caractérise par trois éléments :

- la balise **xgql:function**
- un nom de fonction (identifié par l'attribut **name**)
- un bloc d'instruction

Une procédure peut intéragir avec les variables déclarées en global dans le programme.

```
<xgql:var name="var1">5</xgql:var>
<xgql:function name="func">
<xgql:var name="var1">$var1 - 1</xgql:var>
</xgql:function>
```

L'appel d'une procédure se fait de la façon suivante :

```
<xgql:call name="func"/>
```

Attention, si plusieurs procédures ont le même nom, c'est la dernière procédure avant l'appel qui sera interprété.

6. Les sources de données

Un des principaux avantages d'XGQL, en plus d'accéder à toutes les fonctionnalités XQuery, est de pouvoir gérer simultanément plusieurs sources de données :

- FTP
- http
- FILE

Les données sont accessibles avec l'instruction **xgql:datasource**.

Accès à un document via FTP
```
<xgql:datasource>ftp://username:password@ftp.whatever.c
om/file.zip;type=i </xgql:datasource>
```

Accès à un document via HTTP
```
<xgql:datasource>http://www.symeria.com
</xgql:datasource>
```

accès à un document sur le file system
```
<xgql:datasource>C:/My
Documents/ALetter.html</xgql:datasource>
```

Lors d'une utilisation simple, le flux du datasource sera transcrit directement en résultat du script. Pour travailler sur ce flux, il suffit juste d'encapsuler le **xgql:datasource** dans une déclaration de variable.

6.1. Travailler sur les données récupérées

Bien entendu, il est possible de travailler sur ces datasources en les récupérant dans une variable. Considérons le fichier test.xml dans le répertoire suivant C:\myTest :

```
<?xml version="1.0" encoding="UTF-8"?>
<account id='42'>
<name>Aurel</name>
<firstname>Marc</firstname>
<login>martin_p</login>
<pwd>mypassword</pwd>
</account>
```

Pour récupérer son contenu dans une variable XGQL, il suffit de faire la déclaration suivante :

```
<xgql:var
name="maVariable"><xgql:datasource>C:\myTest\test.xml</
xgql:datasource></xgql:var>
```

Il est alors possible de travailler sur cette variable, donc sur le XML qu'elle représente :

```
<xgql:process>let $doc:= document{$maVariable} return
$doc/account/data(@id)</xgql:process>
```

Dans cette exemple, **xgql:process** va renvoyer 42 comme résultat. Ceci illustre la facilité avec laquelle il est possible d'utiliser des sources externes de données afin de la manipuler avec XGQL !
7. Interroger des bases relationnelles
Depuis la version 2.0 de XGQL, en plus de gérer de nombreuses sources de données, il est également possible de travailler sur plusieurs bases de données relationnelles simultanément (jusqu'à 6).

7.1. Configuration

Le fichier xgql.properties est le fichier de configuration XGQL. C'est dans ce fichier que sont spécifiés les données de connections aux bases de données :

- **idConnect** : nom de la connexion. Ce nom sera utilisé tout au long du script XGQL pour indiqué quel pool de connexion est utilisé
- **className** : nom du driver JDBC à utiliser pour se connecter

- **url** : url d'accès à la base
- **login** : identifiant de la base
- **password** : mot de passe pour accéder à la base

Ces informations sont à configurer autant de fois qu'il y a de bases de données à interroger. Exemple de fichier **xgql.properties.**

```
# La première connexion est celle par défaut
xgql.connectId=myHsql
xgql.className=org.hsqldb.jdbcDriver
xgql.url=jdbc:hsqldb:file:testdb
xgql.login=SA
xgql.password=

# Les connexion suivantes peuvent actuellement aller de 1 à 5
xgql.1.connectId=mySql
xgql.1.className=com.mysql.jdbc.Driver
xgql.1.url=jdbc:mysql://localhost:3308/valkyrie
xgql.1.login=root
xgql.1.password=pwd
```

Ici, la configuration est prévue pour interroger deux bases.

7.2. CREATE, INSERT, UPDATE, DELETE et DROP

L'exécution d'une requête ne retournant pas d'informations (create, insert, update, delete ou drop) se décompose en deux parties :

- déclaration de la requête dans une variable
- exécution de la requête sur une base précise. Par défaut, c'est la première du **xgql.properties** qui est prise en compte

Si ces deux étapes ne sont pas respectées et que la requête est directement inclus dans la balise d'exécution, le script ne pourra pas lancer cette opération.

Création d'une table

```
<xgql:var name="create">CREATE TABLE testXGQL (id integer, name varchar(20))</xgql:var>
<xgql:execute name="create" connectionId="myHsql"/>
```

Insertion de données

```
<xgql:var name="insert1">INSERT INTO testXGQL VALUES (1, 'test 1')</xgql:var>
<xgql:execute name="insert1" connectionId="myHsql"/>
```

Suppression de données

```
<xgql:var name="delete">DELETE FROM testXGQL WHERE id=2</xgql:var>
<xgql:execute name="delete" connectionId="myHsql"/>
```

Mise à jour de données

```
<xgql:var name="delete">UPDATE testXGQL SET id=4, name='test 4' WHERE id=3</xgql:var>
<xgql:execute name="delete" connectionId="myHsql"/>
```

Suppression de la table

```
<xgql:var name="drop">DROP TABLE testXGQL</xgql:var>
<xgql:execute name="drop" connectionId="myHsql"/>
```

7.3. SELECT

7.3.1. Traitement de requête

Une requête de sélection se réalise de la même que les requêtes déjà vues précédemment, par le **xgql:execute**. La gestion des éléments retour se fait par le biais du **xgql:row** et du **xgql:column**.

```
<xgql:var name="select">SELECT * FROM testXGQL</xgql:var>
<xgql:execute name="select" connectionId="myHsql">
<xgql:row>
<xgql:var name="id">
<xgql:column name="id" wrap="false"/>
</xgql:var>
</xgql:row>
</xgql:execute>
```

Le **xgql:row** définit le bloc sur lequel le script va itérer pour chaque tuple résultat de la requête. **xgql:column** indique, via l'attribut name, quel est le nom colonne dont on souhaite obtenir la valeur. L'attribut wrap définit le format de cette sortie. Si wrap est à true, la valeur sera renvoyée dans une balise du nom de la colonne, sinon, c'est uniquement la valeur qui est renvoyé.
Dans l'exemple précédent,

```
<xgql:column name="id" wrap="false"/>
```

renvoie la valeur de l'id, alors que

```
<xgql:column name="id" wrap="true"/>
```

renvoie

```
<id>value</id>
```

En fonction du type de sortie ou du traitement choisi, il est donc possible de générer de simples données textes ou un flux XML depuis une ou plusieurs bases de données. Exemple de sortie en texte :

```
<xgql:var name="select">SELECT * FROM testXGQL</xgql:var>
<xgql:execute name="select" connectionId="myHsql">
<result>
<xgql:row>
<xgql:var name="id"><xgql:column name="id" wrap="false"/></xgql:var>
<xgql:var name="test"><xgql:column name="name" wrap="false"/></xgql:var>
Id is <b>$id</b> and value is <b>$test</b><br/>
</xgql:row>
</result>
</xgql:execute>
```

7.3.2. Branche conditionnelle

Un peu comme le **xgql:if**, le branchement conditionnel est possible sur le **xgql:execute** avec la balise **xgql:norow**. Le bloc d'instruction de cette balise est interprété si la requête ne renvoie aucun résultat.

```
<xgql:var name="select">SELECT * FROM testXGQL</xgql:var>
<xgql:execute name="select" connectionId="myHsql">
<xgql:row>
<xgql:var name="id"><xgql:column name="id" wrap="false"/></xgql:var>
</xgql:row>
<xgql:norow>Pas de resultat</xgql:norow>
</xgql:execute>
```

Il ne peut pas y avoir plusieurs instructions **xgql:row** dans le même **xgql:execute**. Le résultat du xgql:execute est du XML. Les données sont donc pré-formatées.

7.4. Travailler sur les données provenant de la base relationelle

Les bases relationnelles constituent bien sûr un datasource.
Lors de l'exécution d'un SELECT, on constate que le résultat est formaté directement en XML.
Il est donc possible d'encapsuler ce résultat dans une variable et de manipuler le XML obtenu grâce à du XQuery comme nous l'avons vu précédement.
Cela signifie également qu'il est très facile d'interpoler des données provenant des sources de données XML avec les données provenant de sources relationnelles.

8. Les extensions de langage

XGQL reste un langage très ouvert au travers de nombreuses extensions fournies ou non en natif. Par exemple, l'extension web est fournie en natif avec la version 2.0, incluant toute une série d'instructions supplémentaires pour la gestion du web (cookies, sessions, etc.)
Pour inclure de nouvelles extensions, il suffit de le déclarer de la façon suivante :

```
<xgql:extension name="SaxParserExtnExample"
package="my.own.package"/>
```

name est le nom de l'extension et l'attribut package indique le nom du package de l'extension. Par défaut, le nom du package est 'org.symeria.xgql.parser.extension'
Il existe de nombreuses extensions de langage, tel que XGQL:DB2XML qui va rendre transparente la gestion de données XML dans un base de données relationnelles.

9. Les librairies

Nous avons vu précédemment qu'il était possible de réaliser des procédures pour simplifier et factoriser le code. Dans un souci de rentabilité, il est assez possible d'importer des librairies XGQL :

```
<xgql:datasource>C:Documents and
Settings/yourpath/SWAS_XGQL/webapps/XGQL/myLib.xgq</xgq
l:datasource>
```

Les librairies peuvent être accédées aussi par http et ftp.

10. XGQL et le web

XGQL dans sa version SWAS (API XGQL, serveur Jetty et base Hsql, le tout préconfiguré pour fonctionner en servlet) s'adapte au fonctionnalité web, avec la gestion :

- des cookies
- des requests
- des sessions

Néanmoins, l'utilisation de XGQL sous cette forme n'est pas la plus optimum et n'est pas recommandée pour la réalisation de sites complexes ou professionnels.

10.1. Cookies

Création d'un cookie :

```
<xgql:cookie id="test"
expiration="100000">MyCookie</xgql:cookie>
```

Récupération de la valeur d'un cookie

```
<xgql:cookie id="test"/>
```

A ce niveau, la valeur est juste lue. Pour être traitée, elle doit être placée dans une variable.

10.2. Requests

Récupération d'une valeur passée en reqests

```
<xgql:request name="aRequestKeyName"/>
```

10.3. Sessions

Création d'une valeur en session

```
<xgql:session name="aSessionKeyName">My session
Value</xgql:session>
```

Récupération de la valeur en session

```
<xgql:session name="aSessionKeyName"/>
```

A ce niveau, la valeur est juste lue. Pour être traitée, elle doit être placée dans une variable.

11. Conclusion

En conclusion, il n'y aura plus qu'une chose à préciser, mis à part ce petit détail : "Où trouver les dernières distributions XGQL ?"
Sur le site officiel XGQL est à l'adresse suivante Lien51. Différentes versions de l'API XGQL y sont disponibles (API, source, SWAS, etc.) ainsi que de nombreux exemples.

Retrouvez la suite de l'article de Rémi Masson en ligne : Lien52

Les derniers tutoriels et articles

Tutoriel : Développement Dirigé par les Tests

Cette page présente un tutoriel sur le développement dirigé par les tests (ou Test Driven Development en anglais). Le tutoriel présente les principes de cette méthode, et ensuite l'illustre pas à pas sur un exemple concret : la recherche de toutes les solutions du problème des pentaminos. Les principaux outils utilisés sont Visual C# Express et Nunit. Les notions de couverture de code, de complexité cyclomatique sont également abordées.

1. Introduction

Ce tutoriel propose la mise en oeuvre d'un développement dirigé par les tests (Test Driven Development - TDD) sur un cas concret selon les principes exposés par Kent Beck dans son livre Test-Driven Development: By Example (Lien53).

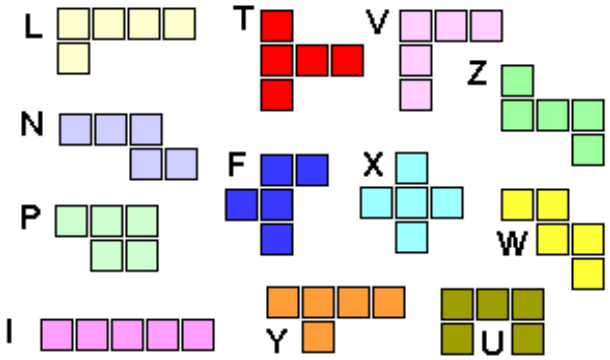
L'exemple donné par Kent Beck (un calculateur multi-monnaies) est trompeusement simple, si bien que des lecteurs sous-estiment parfois l'intérêt du TDD. Aussi ce tutoriel s'appuie sur un sujet qui paraîtra plus difficile, afin de mieux montrer l'apport du TDD.

Par rapport à d'autres tutoriels qui existent déjà, comme

- Tests unitaires (Lien54)
- Les tests unitaires avec Nunit (Lien55)

nous allons essayer de montrer la mise en pratique de cette démarche de A à Z, afin de bien insister sur la discipline particulière que propose Kent Beck.

Le sujet que nous allons traiter est le calcul de toutes les solutions possibles du jeu des pentaminos (voir Lien56).



L'objectif est de placer ces douze pentaminos sur un rectangle de 6 par 10 cases, sans aucun trou ni chevauchement. Grâce à wikipedia, nous savons d'avance qu'il y a 2339 solutions uniques (4 fois plus en comptant les solutions symétriques), et dans ce tutoriel nous souhaitons pouvoir les afficher toutes.

Réaliser un tel programme paraît souvent difficile, notamment en raison du caractère "géométrique" du problème, et c'est à ce titre qu'il nous intéresse comme support d'introduction du TDD. De plus ce problème n'est pas uniquement ludique, c'est un cas particulier d'un problème de satisfaction de contraintes qu'un développeur peut être amené à rencontrer ; il est donc intéressant d'en connaître les principes de résolution.

Le projet sera réalisé en Visual C# 2005 Express Edition, téléchargeable gratuitement sur le site de Microsoft (Lien57) et avec l'aide de l'outil de tests unitaires Nunit, également téléchargeable gratuitement (Lien58 - prendre la version win .net 2.0).

2. Principes du TDD

L'objectif du TDD est de produire du "code propre qui fonctionne". Pour cela, deux principes sont mis en oeuvre :

- un développeur écrit du code nouveau seulement lorsqu'un test automatisé a échoué
- toute duplication de code (ou plus généralement d'information, ou de connaissances) doit être éliminée. L'acronyme anglais DRY (Do not Repeat Yourself) peut être utilisé comme moyen mnémotechnique pour cette phase très importante.

Ces deux principes doivent être strictement respectés, même s'ils paraissent difficiles ou bizarres dans un premier temps. Bien comprendre le TDD suppose en effet de respecter strictement la discipline imposée par le cycle décrit ci-dessous.

Bien que simples, ils ont diverses implications :

- nous allons concevoir notre code de manière incrémentale, en ayant toujours du code en état de marche, de telle sorte que ce code nous fournisse de l'information pour prendre de nombreuses petites décisions au cours de notre développement.
- nous devons écrire nos propres tests, parce que nous ne pouvons pas attendre de nombreuses fois par jour qu'une autre personne le fasse
- notre environnement de développement doit fournir une réponse ultra rapide en cas de petits changements
- notre code doit être composé d'éléments très cohérents et très peu couplés, afin de rendre le test facile.

Le travail se fait en trois phases. Les deux premières sont nommées d'après la couleur de la barre de progrès dans les outils comme Nunit :

- ROUGE: écrire un petit test qui échoue, voire même ne compile pas dans un 1er temps
- VERT : faire passer ce test le plus rapidement possible, en s'autorisant si besoin les "pires" solutions : S'il existe une solution propre, simple et immédiate, réalisez-la Si une telle solution prend plus d'une minute, notez la et revenez au problème principal : avoir une barre de progrès verte en quelques secondes
- REMANIEMENT (refactoring en anglais) : éliminer absolument toute duplication apparue durant les étapes 1 et 2.

Pour réaliser l'étape 2 ci-dessus, il y a trois stratégies :

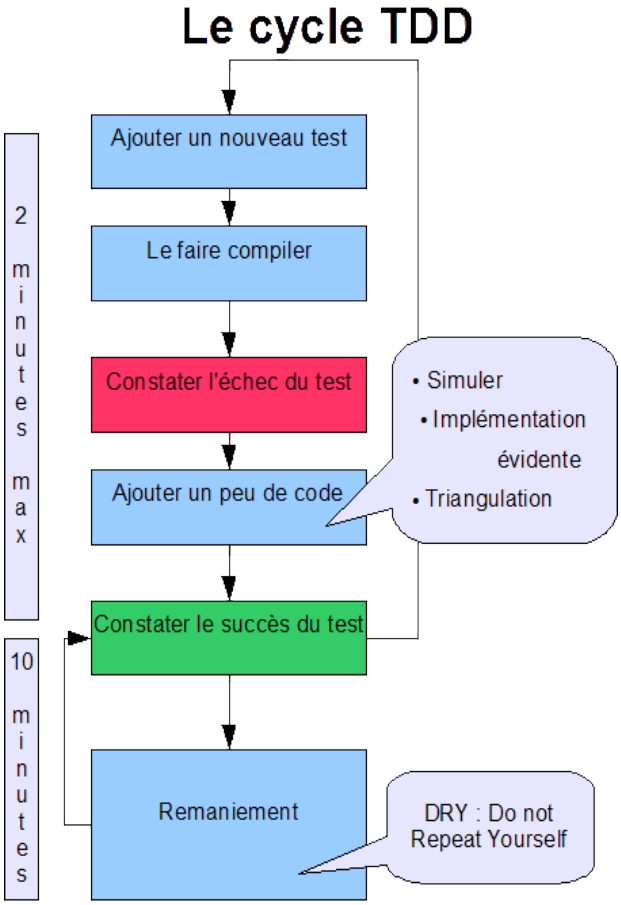
- Simulation : retourner une constante, puis remplacer progressivement ces constantes avec des variables afin d'obtenir le code réel
- Implémentation évidente : taper directement la bonne solution
- Triangulation : avoir deux exemples du résultat recherché, et généraliser.

Le cycle de travail en TDD est donc le suivant :

- ajouter rapidement un nouveau test
- Exécuter tous les tests, et constater l'échec du nouveau : ROUGE
- Faire un petit changement
- Exécuter tous les tests, et constater qu'ils passent : VERT
- Remanier le code pour éliminer toute duplication : REMANIEMENT

Il est essentiel que ce cycle se réalise très rapidement, en quelques minutes tout au plus. Si la réalisation du cycle prend des dizaines de minutes, il est probable que vous soyez en train d'essayer de réaliser un pas trop important ; il est alors souhaitable d'essayer d'attaquer une étape moins ambitieuse. Comme nous le verrons, la particularité du TDD est de réaliser des étapes qui peuvent paraître minuscules à des développeurs chevronnés, certains parlent même de micro-incréments de code.

Le cycle complet est résumé par le diagramme ci-dessous, avec l'ordre de grandeur du temps à consacrer à chaque phase.



3. Premiers pas

3.1. Mise en place du projet

Après avoir téléchargé et installé Visual C# Express et Nunit, nous créons une application console (Menu Fichier, Nouveau Projet, et Application Console) appelées Pentaminos. Ensuite il faut ajouter la référence Nunit-framework au projet, depuis l'explorateur de solution Visual C#.

Sauvegarder la solution (menu Fichier, Sauver Tous) puis la construire (F6).

Lancer Nunit GUI, puis ouvrir l'executable produit par Visual C#, il se trouvera dans Pentaminos\bin\Release\Pentaminos.exe. Nunit GUI affiche alors notre projet, avec aucun test pour l'instant.

A partir de là nous allons passer constamment de Visual C# à Nunit : après de petites modifications dans le code C# (et reconstruction de la solution par F6) nous passerons dans Nunit GUI pour appuyer sur le bouton Run, et examiner la couleur de la barre de progression située en dessous de Run.

3.2. Premier test

Afin de vérifier le fonctionnement de nos outils, ajoutons un test élémentaire. Il s'agit de vérifier que le programme peut retourner une description. Les principes du TDD nous imposent d'écrire le test d'abord. Donc nous ajoutons dans le fichier Program.cs le code suivant :

```
[TestFixture]
public class TestProgram{
   [Test]
   public void TestDescription(){
       Assert.That(Program.Description.Length,
Is.GreaterThan(0));
   }
}
```

Nous remarquons ci-dessus plusieurs éléments apportés par Nunit:

- l'attribut TestFixture, qui permet d'indiquer qu'une classe est une classe de test, et qu'elle sera visible par Nunit
- l'attribut Test, qui permet d'indiquer qu'une méthode d'une classe de test est un test. Chacune de ces méthodes est exécutée de manière complètement indépendante: en effet, Nunit va créer une nouvelle instance de la classe de test pour exécuter chacune des méthodes de tests.
- Une assertion, sous la forme Assert.That(,). Dans ce tutoriel nous utilisons une forme très récente de ces assertions, introduite dans Nunit 2.4. Celle nouvelle forme est plus lisible et plus évolutive que la forme dite classique, dans laquelle l'assertion aurait pris la forme :

```
Assert.Greater(Program.Description.Length,0) ;
```

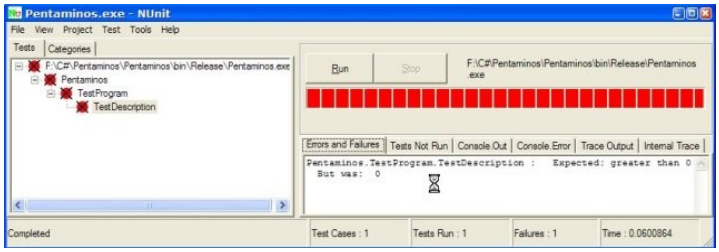
Et il faut également insérer les déclarations suivantes au début du fichier :

```
using Nunit.Framework;
using Nunit.Framework.Constraints;
using Nunit.Framework.SyntaxHelpers;
```

A ce stade, notre code ne compile pas, ce qui est prévisible. Il faut ajouter une propriété description à la classe Program :

```
static public string Description{
   get { return ""; }
}
```

C'est le minimum que nous pouvons faire pour pouvoir compiler. Nunit peut alors nous montrer l'exécution de ce test, qui bien sûr échoue, et donc la barre est ROUGE :



Observez également que Nunit donne des informations intéressantes sur la cause de l'échec du test.

Le minimum pour faire passer le test est de retourner la valeur attendue :

```
get { return "Pentaminos"; }
```

La barre Nunit devient alors verte.

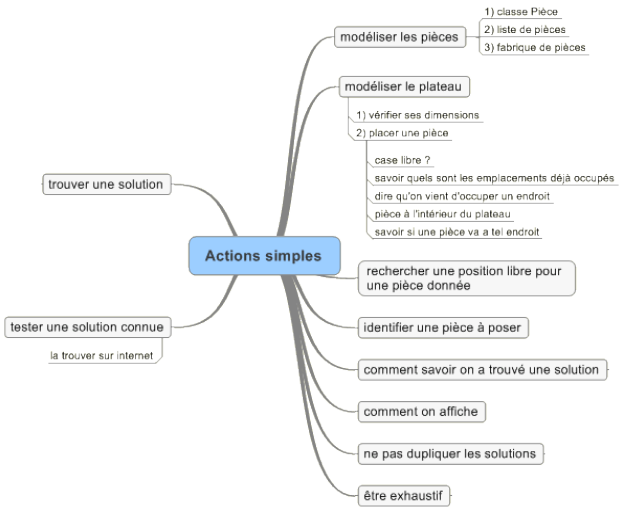
La dernière étape du cycle TDD nous impose de supprimer toute duplication de code ; il n'y en a pas ici, donc nous avons fini le cycle.

Avec cet exemple élémentaire, nous avons donc pu :

- vérifier le fonctionnement de nos outils
- observer un cycle complet de TDD : compilation, barre ROUGE, barre VERTE, remaniement du code

3.3. Notre liste de travail initiale

A ce stade, nous ne savons pas nécessairement comment programmer la recherche des solutions, mais cela ne doit pas nous bloquer. En effet Kent Beck recommande de mettre une point une liste de toutes les actions de programmation simples et pertinentes que nous pouvons imaginer pour progresser sur ce projet. Voici une liste de départ élaborée en quelques minutes de réflexion :



Ici nous avons utilisé un format particulier (une "mind map"), mais n'importe quel support peut être utilisé, à commencer par le papier ou un tableau blanc.

Partant de cette liste, nous pouvons commencer à travailler en TDD sur l'action qui nous paraît la plus élémentaire possible (rappelez-vous que nous allons chercher à faire des micro-incréments de code). Commençons par la modélisation des pentaminos.

4. Modélisation des pentaminos

En tenant compte de toutes les rotations et symétries possibles, les 12 pentaminos peuvent présenter en tout 63 variantes. Nous allons simplement réutiliser une modélisation de ces pentaminos déjà faite par David Eck (avec sa permission), sous la forme suivante :

```
{
          { 1, 1,2,3,4 },          // This array
represents everything the program
          { 1, 10,20,30,40 },      // knows about the
individual pentominos.  Each
          { 2, 9,10,11,20 },       // row in the array
represents a particular
          { 3, 1,10,19,20 },       // pentomino in a
particular orientation.  Different
          { 3, 10,11,12,22 },      // orientations are
obtained by rotating or flipping
          { 3, 1,11,21,22 },       // the pentomino
over.  Note that the program must
          { 3, 8,9,10,18 },        // try each pentomino
in each possible orientation,
          { 4, 10,20,21,22 },      // but must be
careful not to reuse a piece if
          { 4, 1,2,10,20 },        // it has already
been used on the board in a
          { 4, 10,18,19,20 },      // different
orientation.
          { 4, 1,2,12,22 },        //     The
pentominoes are numbered from 1 to 12.
          { 5, 1,2,11,21 },    // The first number on
each row here tells which pentomino
          { 5, 8,9,10,20 },        // that line
represents.  Note that there can be
          { 5, 10,19,20,21 },      // up to 8 different
rows for each pentomino.
          { 5, 10,11,12,20 },      // some pentominos
have fewer rows because they are
          { 6, 10,11,21,22 },      // symmetric.  For
example, the pentomino that looks
          { 6, 9,10,18,19 },       // like:
          { 6, 1,11,12,22 },       //           GGG
          { 6, 1,9,10,19 },        //           G G
          { 7, 1,2,10,12 },        //
          { 7, 1,11,20,21 },       // can be rotated
into three additional positions,
          { 7, 2,10,11,12 },       // but flipping it
over will give nothing new.
          { 7, 1,10,20,21 },       // So, it has only 4
rows in the array.
          { 8, 10,11,12,13 },      //     The four
remaining entries in the array
          { 8, 10,20,29,30 },      // describe the given
piece in the given orientation,
          { 8, 1,2,3,13 },         // in a way
convenient for placing the piece into
          { 8, 1,10,20,30 },       // the one-
dimensional array that represents the
          { 8, 1,11,21,31 },       // board.  As an
example, consider the row
          { 8, 1,2,3,10 },         //
          { 8, 10,20,30,31 },      //           { 7,
1,2,10,19 }
          { 8, 7,8,9,10 },         //
          { 9, 1,8,9,10 },         // If this piece is
placed on the board so that
          { 9, 10,11,21,31 },      // its
topmost/leftmost square fills position
          { 9, 1,2,9,10 },         // p in the array,
then the other four squares
          { 9, 10,20,21,31 },      // will be at
positions  p+1, p+2, p+10, and p+19.
          { 9, 1,11,12,13 },       // To see whether the
piece can be played at that
          { 9, 10,19,20,29 },      // position, it
suffices to check whether any of
          { 9, 1,2,12,13 },        // these five squares
are filled.
          { 9, 9,10,19,29 },
```

```
            { 10, 8,9,10,11 },
            { 10, 9,10,20,30 },
            { 10, 1,2,3,11 },
            { 10, 10,20,21,30 },
            { 10, 1,2,3,12 },
            { 10, 10,11,20,30 },
            { 10, 9,10,11,12 },
            { 10, 10,19,20,30 },
            { 11, 9,10,11,21 },
            { 11, 1,9,10,20 },
            { 11, 10,11,12,21 },
            { 11, 10,11,19,20 },
            { 11, 8,9,10,19},
            { 11, 1,11,12,21 },
            { 11, 9,10,11,19 },
            { 11, 9,10,20,21 },
            { 12, 1,10,11,21 },
            { 12, 1,2,10,11 },
            { 12, 10,11,20,21 },
            { 12, 1,9,10,11 },
            { 12, 1,10,11,12 },
            { 12, 9,10,19,20 },
            { 12, 1,2,11,12 },
            { 12, 1,10,11,20 }
        };
//          by:  David J. Eck
//               Deparment of Mathematics and Computer
Science
//               Hobart and William Smith Colleges
//               Geneva, NY   14456
//               Email:  eck@hws.edu
//
```

Toutefois, le TDD nous interdit d'écrire du code avant d'avoir un test. Quel test écrire ici ? Eh bien nous pourrions vérifier qu'il y a bien 63 éléments en tout (une erreur de copier/coller est toujours possible). Donc le test pourrait être

```
Assert.That(ListeDePentaminos().Count, Is.EqualTo(63));
```

En fait il faut que cette liste soit portée par une classe, donc nous choisissons d'introduire une classe FabriqueDePentaminos comme ci-dessous:

```
Assert.That(FabriqueDePentaminos.ListeDePentaminos().Co
unt, Is.EqualTo(63));
```

Le test étant défini, nous avons maintenant la permission d'écrire du code : ajoutons un fichier Pentaminos.cs au projet, fichier qui contiendra la classe de test

```
[TestFixture]
public class TestListePentaminos{
   [Test]
   public void TestTotalPentaminos(){
      Assert.That(FabriqueDePentaminos.ListeDePentamino
s().Count, Is.EqualTo(63));
   }
}
```

ainsi que tout ce qui concernera la définition des pentaminos et de leur fabrique.

Voici le code minimum pour parvenir à compiler, et passer à la barre ROUGE :

```
class Pentamino{}

class FabriqueDePentaminos{
    static public List<Pentamino> ListeDePentaminos(){
        return new List<Pentamino>();
    }
}
```

Pour avoir la barre verte, on peut écrire le code suivant :

```
static public List<Pentamino> ListeDePentaminos(){
        List<Pentamino> liste = new List<Pentamino>();
        for (int i = 0; i < 63; i++){
                liste.Add(new Pentamino());
        }
        return liste;
}
```

Attention, 63 est dupliqué ! La troisième phase du cycle nous impose de supprimer toute duplication, ce que nous pouvons faire en introduisant une constante portée par FabriqueDePentaminos :

```
public const int NombreDeVariantes = 63;
```

Le test suivant va nous forcer à remplir effectivement la structure de Pentomino. Par exemple, vérifions que le 3ème élément de la liste est bien le pentamino X, si nous avons bien compris la modélisation ; le test est alors :

```
[Test]
public void TestPositionDuPentominoX(){
   Pentamino x =
FabriqueDePentaminos.ListeDePentaminos()[2] ;

   Assert.That(x.Decalage[0], Is.EqualTo(9),
       "le premier décalage de X est incorrect");
   Assert.That(x.Decalage[1], Is.EqualTo(10),
       "le deuxième décalage de X est incorrect");
   Assert.That(x.Decalage[2], Is.EqualTo(11),
       "le troisième décalage de X est incorrect");
   Assert.That(x.Decalage[3], Is.EqualTo(20),
       "le quatrième décalage de X est incorrect");
   Assert.That(x.Variante, Is.EqualTo(2),
       "la Variante de X est incorrecte");
}
```

Ce test comprend plusieurs assertions à la suite : ceci n'est pas souhaitable en général, pour les raisons suivantes :

- en cas d'échec, il est plus difficile d'identifier quel test a échoué
- le test s'arrête dès que l'une des assertions échoue, ce qui nous prive d'informations complémentaires sur le code qui est testé ; en pratique, dans des cas plus compliqués, nous serons souvent obligés de commenter l'assertion qui a échoué, afin de voir si les autres assertions passent ou pas.

Ici nous choisissons tout de même grouper ces 5 assertions dans un seul test, et nous rendons plus évident le test qui échouera en mettant une chaîne de caractères dans l'assertion.

Ce test nous conduit à définir l'interface de Pentamino comme suit :

```
public int[] Decalage = new int[4];
public int Variante;
```

ce qui nous permet d'arriver à la barre rouge. Pour avoir la barre verte, il faut maintenant remplir les pentaminos avec leur description, donc :

- ajouter un constructeur,

- insérer un tableau à deux dimensions qui contiendra la description interne des pentaminos (celle obtenue de D. Eck)
- parcourir ce tableau dans la méthode ListeDePentaminos

Ces portions ne code ne sont pas reproduites ici, voir directement le fichier source Pentamino.cs.

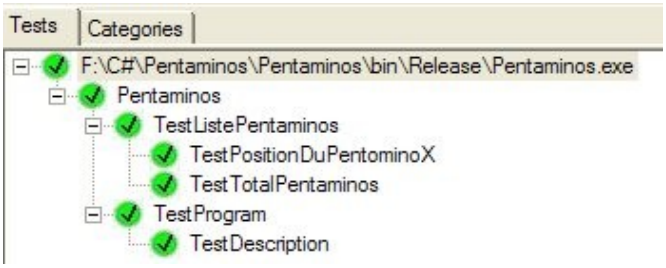
On peut alors remarquer qu'il est possible de remplacer la constante 63 par la dimension du tableau interne, de la manière suivante :

```
static public int NombreDeVariantes{
    get { return DescriptionsInternes.GetUpperBound(0)
+ 1; }
}
```

Ceci permet les remarques suivantes :

- ce changement peut être fait en toute sécurité, car les tests passent toujours tous
- nous avons éliminé une autre duplication, moins évidente. Plus tard s'il faut modifier le nombre de variantes de pentaminos, il suffirait d'agir à un seul endroit, le tableau des descriptions internes.

A ce stade nous avons une première modélisation des pentaminos, couverte par deux tests qui sont présentés ci-dessous :



Et nous pouvons avancer sur le point suivant, la modélisation du plateau qui accueillera les pentaminos.

5. Modélisation du plateau

Il y a de nombreuses manières d'aborder ce point ; par exemple il est tentant de réfléchir à la structure interne de ce plateau. Allons-nous utiliser un tableau à deux dimensions, ou bien à une dimension comme la description interne des pentaminos le suggère ? Laquelle sera la plus pratique ? La plus efficace ?

Toutefois nous devons nous laisser guider par des tests avant de coder quoi que ce soit. Imaginer de tels tests est souvent très difficile pour les débutants en TDD ; pour les aider, une autre manière de voir les tests est d'imaginer qu'ils représentent des **exemples** de ce que l'on veut faire. Ceci va nous conduire à définir en premier une interface (un contrat) plutôt qu'une structure interne.

Voici des exemples de ce que nous souhaitons faire avec le plateau :

- pouvoir ajouter un pentamino
- ne pas pouvoir ajouter deux fois le même pentamino
- ne pas permettre de chevauchement de pentamino
- les pentaminos ajoutés ne devront pas déborder du plateau
- savoir si une solution a été trouvée
- pouvoir enlever un pentamino
- pouvoir afficher un plateau (en mode console)

5.1. Premiers tests

Commençons donc par écrire un test, qui bien sûr ne compilera même pas :

```
Assert.That(plateau.Ajoute(I));
```

pour traduire l'intention d'ajouter un pentamino. Pour compiler, il faut alors ajouter une nouvelle classe Plateau (et un nouveau fichier Plateau.cs au projet), et le code suivant :

```
class Plateau{
    public Boolean Ajoute(Pentamino pentamino){
        return false;
    }
}

[TestFixture]
public class TestPlateau{
    [Test]
     public void TestAjoutPentamino(){
        Plateau plateau = new Plateau();
        Pentamino I =
FabriqueDePentaminos.ListeDePentaminos()[0] ;
        Assert.That(plateau.Ajoute(I));
    }
}
```

Ce qui permet de compiler et d'arriver à la barre rouge. Le minimum pour arriver à la barre verte est alors de changer false en true dans la méthode Ajoute.

Ces étapes minimalistes peuvent paraître superflues. Il n'en est rien. Ces petites étapes permettent de valider l'outil de test avant d'écrire le véritable code de production. En effet sur un projet réel, qui comportera des milliers de tels petits tests, il n'y a rien de pire qu'un test qui est vert tout de suite : il peut être vert par hasard, ou par erreur de construction (la condition de l'assertion est toujours vérifiée). On peut également être en train de travailler sur un autre test que celui que l'on imagine... En d'autres termes, ce qui valide un test, ce n'est pas la barre verte, c'est l'observation du passage de la barre rouge à la barre verte.

Ayant notre barre verte, et n'ayant a priori pas introduit de duplication, il faut avoir un autre test avant d'écrire plus de code ! Tout simplement

```
[Test]
public void TestAjoutPentaminoSansRepetition(){
        Plateau plateau = new Plateau();
        Pentamino I =
FabriqueDePentaminos.ListeDePentaminos()[0];
        plateau.Ajoute(I);
        Assert.That(plateau.Ajoute(I), Is.False);
}
```

Ce code compile, et donne la barre rouge comme attendu. Il introduit également des duplications dans le code de test, point à régler dès que la barre verte sera obtenue. Pour obtenir la barre verte, impossible de continuer à changer des "true" en "false" : ce cas est un exemple de triangulation, où nous sommes contraints par deux tests au moins à écrire du code (ce qui oblige à généraliser).

Ici une solution assez simple est possible, donc écrivons-là directement pour avoir la barre verte :

```
private Boolean[] VariantesDejaAjoutees = new
Boolean[12] ;
```

```
public Boolean Ajoute(Pentamino pentamino){
        if (VariantesDejaAjoutees[pentamino.Variante]){
                return false;
        } else {

        VariantesDejaAjoutees[pentamino.Variante] =
true;
                return true;
        }
}
```

Il faut maintenant passer à la duplication de code présente dans la classe de test, où les deux méthodes comportent des initialisations communes. Nunit permet de grouper ces initialisations dans une méthode Setup comme suit :

```
private Plateau plateau;
private Pentamino I;

[SetUp]
public void SetUp(){
   plateau = new Plateau();
   I = FabriqueDePentaminos.ListeDePentaminos()[0];
}
```

Cette méthode SetUp est automatiquement appelée par Nunit avant l'exécution de chaque test (tout comme une méthode TearDown est appelée après, afin de libérer des resources si besoin).

Cela permet de supprimer la duplication, comme d'habitude en toute sécurité puisque la barre reste verte après cette opération.

A ce stade nous avons confiance que la méthode Ajoute gérera correctement les ajouts de pentaminos sans permettre d'ajouter deux fois le même, ni deux variantes du même pentamino, donc nous n'écrivons pas plus de tests sur ce point.

Maintenant nous pouvons déjà traiter l'identification d'une solution trouvée, en considérant qu'il suffira qu'un pentamino de chaque sorte ait été posé.

Voici un premier test

```
[Test]
public void TestSolutionTrouveePlateauVide(){
    Assert.That(plateau.SolutionTrouvee, Is.False);
}
```

pour compiler et passer à la barre rouge :

```
public Boolean SolutionTrouvee{
    get { return true; }
}
```

Pour arriver à la barre verte, il y a deux options :

a) Changer true en false (barre verte) - mais cette solution ne marche pas dans tous les cas, il faut alors ajouter un test supplémentaire, du type ajouter les 12 pentaminos d'une solution connue, et vérifier la valeur de SolutionTrouvee. C'est la méthode de triangulation, qui continue à nous faire faire de tout petits pas.

b) Considérer que l'implémentation est évidente, et sans risque, et faire un pas un peu plus grand.

Prenons l'implémentation évidente, afin de montrer que le développeur a le choix de la taille des pas :

```
public Boolean SolutionTrouvee{
     get { return (TotalVariantesDejaAjoutees ==
FabriqueDePentaminos.NombreDePentaminos) ; }
}
```

avec de plus :

- le code nécessaire à la déclaration et l'initialisation de TotalVariantesDejaAjoutees
- l'incrémentation de TotalVariantesDejaAjoutees dans la méthode Ajoute
- l'extraction d'une méthode privée Pose, afin de rendre indissociables les deux opérations qu'il faut maintenant faire quand un pentamino est ajouté. Noter que nous ne cherchons pas à tester cette méthode privée : en TDD on se contente de tester l'interface publique d'une classe, afin d'éviter que le test ne se fragilise en devenant dépendant d'une structure interne susceptible d'évoluer.
- la déclaration d'une constante NombreDePentaminos dans la fabrique de pentaminos, afin de supprimer la duplication du nombre 12.

Voici donc un pas plus important, qui n'est pas sans risque ; un débutant programmeur aurait certainement intérêt à prendre l'option a) ci-dessus.

Retrouvez la suite de article Bruno Oriser en ligne et découvrez la solution de ce problème : Lien59

Les derniers tutoriels et articles

Création et utilisation de DLL dans mIRC

Cet article va vous permettre de réaliser une DLL qui sera utilisable dans mIRC. Vous trouverez le code d'un DLL en PureBasic et en C/C++. Ensuite, vous apprendrez comment faire pour utiliser les fonctions de la DLL dans mIRC. Cette solution va permettre d'étendre les possibilités de mIRC.

1. Introduction

Comme beaucoup d'autres langages intégrés, le mIRC scripting ne permet pas de faire tout ce que l'on désire. En effet le mIRC script permet de faire beaucoup d'actions sur la gestion des évènements IRC mais dès que l'on veut faire quelque chose d'un peu compliqué on est vite freiné.

Pour palier à ces petits inconvénients, mIRC dispose de différentes solutions, à savoir le DDE (communication interprocessus) et le couplage avec une DLL. Nous allons ici parler des DLLs qui permettent d'ouvrir un monde infini sur les possibilités d'options de vos scripts.

Qu'est-ce qu'une DLL (Dynamic Link Library) ?

Microsoft définie une DLL comme un module qui contient des données et des fonctions. Une DLL est chargée en mémoire par un programme appellant (.exe) ou bien par une autre bibliothèque (.dll). En ce sens, il y a deux types de fonctions dans une librairie, les internal et external.

- **external** : ces fonctions peuvent être appellées par n'importe quel module.
- **internal** : ces fonctions ne sont accessibles que part le module appellant.

2. Utilisation de DLL dans mIRC

Avant de développer une DLL, il serait quand même judicieux de savoir comment faire pour utiliser celle-ci. Il n'y a besoin d'aucunes manipulations pour charger une DLL dans mIRC, le simple fait de l'appeler par la fonction du script la charge automatiquement.
Il y a deux possibilités d'appeler une DLL :

Par appel de commande basique
```
/dll <nomDeLaDLL> <fonction> <paramètres>
```

Par fonction de scripting
```
$dll(<nomDeLaDLL>, <fonction>, <paramètres>)
```

Attention : il y a une différence fondamentale entre ces deux appels. Le premier ne permet pas de récupérer la valeur de retour, tandis que l'autre oui.
Comment récupérer la (les) valeur(s) retournée(s)

```
var %result =
$dll(maDLL.dll,maJolieFonctionQuiRetourne,parametres)
```

Ensuite concernant la mise en mémoire de la DLL, soit elle reste en mémoire soit elle attend que mIRC la décharge. Au bout de 10 min d'inactivité, mIRC essaye de décharger la DLL de la mémoire. Encore une fois, c'est la DLL qui va décider de ce qu'elle veut faire.
Voici un bout de code très connu qui permet de lister toutes les DLL chargées.

Tappez
```
//var %c = $dll(0) | while %c { echo -a $dll(%c) | dec
%c }
```

Si vous voulez forcer une DLL à être déchargée, utilisez l'option -u. Pour décharger toutes les DLL :

Tappez
```
//var %c = $dll(0) | while %c { dll -u $dll(%c) | dec
%c }
```

En ce qui concerne les conventions de nommage, il n'en existe pas vraiment ... Tout cela est laissé à l'appréciation du développeur. [Petite parenthèse : je profite de ce moment pour vous conseiller de bien réfléchir lors de la conception de votre DLL pour les noms des fonctions de celle-ci]

3. Création de la DLL

3.1. Généralités et bases

3.1.1. Prototype

Voici le pseudo code de l'entête à utiliser pour faire communiquer la DLL avec mIRC :

```
fonction(HWND mWnd, HWND aWnd, char *data, char
*params, BOOL print, BOOL nopause)
```

- **fonction** est le nom de la fonction que nous allons utiliser sous mIRC (ATTENTION à la casse !)
- **HWND mWnd** est le "handle" de la fenêtre principale de mIRC (c'est l'identifiant de la fenêtre mIRC)
- **HWND aWnd** est le "handle" de la fenêtre active (en général NULL car utilisé par un script)
- **char *data** contient le pointeur vers les données que l'on envoie à la DLL (900 octets max)
- **char *params** contient les paramètres pour exécuter une commande mIRC (900 octets max)
- **BOOL print** contient FALSE si appelé par /.dll pour cacher la commande, sinon contient TRUE

- **BOOL nopause** si ce paramètre contient TRUE, alors mIRC exécute une routine critique.

3.1.2. Valeurs de retour

Les valeurs de retour de fonction sont limitées et chacunes influent sur le comportement de mIRC.

- **0** Arrête l'exécution du script (/halt)
- **1** mIRC peut continuer à traiter le script
- **2** data est rempli avec une commande et params avec les paramètres
- **3** data contient ce que $dll() doit afficher

3.2. Implémentation PureBasic

Créez un projet de DLL, et insérez le code suivant :

Squelette d'une procédure de création de DLL pour mIRC

```
ProcedureDLL HelloWorld(mWnd, aWnd, *StringData,
*StringParam, show.b, nopause.b)
  ;ici on récupère les parametres envoyer via $dll()
  host$ = PeekS(*StringData)

  ;ici on place la chaine dans le tube de sortie
  PokeS(*StringData, "HelloWorld ! Arguments DLL :" +
host$)

  ; on retourne 3 car il a un retour dans *StringData
  ProcedureReturn 3
EndProcedure
```

On compile ce code sous hello.dll, et l'on place sa DLL dans le répertoire courant de mirc.exe. En écrivant l'alias suivant :

A mettre dans Alias

```
/testDLL {
    set %param coucou!
    echo -a $dll(hello.dll, HelloWorld, %param)
}
```

On devrait voir apparaître sur la fenetre active :
HelloWorld ! Arguments DLL :coucou!

3.3. Implémentation C++

Créer un projet de DLL et ajouter un fichier DEF à votre solution afin de pouvoir exporter les différentes fonctions de votre composition.
Cet exemple a été fait avec VS, je mettrai à votre disposition plus tard un autre exemple pour DevC++.

3.3.1. Fichier CPP

Dans ce fichier nous allons mettre nos différentes fonctions de notre DLL.

Squelette d'une procédure de création de DLL pour mIRC

```
int __stdcall WINAPI HelloWorld(HWND mWnd, HWND aWnd,
char *data, char *parms, BOOL print, BOOL nopause)
{
    char *args;
    args = malloc(sizeof(char) * strlen(data) + 1);

    strcpy(args, data); // copie des arguments dans une
variable tampon
    strcpy(data,strcat("HelloWorld ! Arguments : ",
args)); // copie dans le tube de sortie

    return 3; // on annonce à mIRC qu'il y a des infos
à retourner
}
```

3.3.2. FIchier DEF

Attention, votre fichier DEF doit comporter le même nom que celui de la DLL.
Ce fichier DEF va permettre d'exporter l'ensemble des fonctions de votre librairie. Vous devez mettre chaque fonction que vous mettrez dans votre DLL dans ce fichier, sous la forme :

```
LIBRARY NomDeLaDLL
EXPORTS
fonction1
fonction2
...
```

Dans notre exemple, il faudrait donc rajouter :

```
LIBRARY hello
EXPORTS
HelloWorld
```

4. Conclusion

Nous voici arrivé à la fin de notre article. Vous avez donc comprit qu'il est très simple de fabriquer une DLL pour optimiser les performances et faire des choses que ne permet pas le mIRC script.
Un des exemples classiques est la gestion de la 3D pour les intros, ou encore les possibilités graphiques (docking de popups, ...).
J'espère que cet article vous aura intéressé, n'hésitez pas à me faire des commentaires sur le forum ou à poser des questions sur vos développements.

Retrouvez l'article de Bastien Pino en ligne : Lien60

La FAQ mIRC

Qu'est ce qu'une table de hachage ?

Une table de hachage est un tableau qui à une clé (typiquement une chaîne de caractère) associe une valeur.
Un exemple s'impose : Imaginons que je veuille stocker la date de naissance de tous mes amis. Et bien sans les tables de hachage, laissez moi vous dire que vous avez intérêt à ne pas avoir beaucoup d'amis.
C'est simple, vous avez deux solutions : soit l'on crée une variable pour chaque personne (%agesophie, %ageeric...), soit on fait une variable globale contenant toutes les dates et puis on fait appel aux tokens.

Pour le second cas, vous avez intérêt à avoir une bonne mémoire pour vous rappeler que la date de naissance de machin est en troisième position, celle de truc en 5ème...
Pour le premier cas, vous avez compris le problème...

La solution, c'est la table de hachage ! Voici comment ça va se présenter :

```
Sophie -> 23 12 88
Eric -> 12 03 87
Loïc -> 03 02 85
...
```

Simple et efficace ! Un autre avantage ? Vous ne trouverez pas dans mIRC un moyen plus rapide pour accéder à une donnée ! Pourquoi ? Parce que les tables de hachage sont chargées dans la mémoire vive (très rapide).

Ainsi leur utilisation accélère le fonctionnement de votre script. Je vous recommande donc d'user et d'abuser de ces petites merveilles.

Comment créer et accéder à une table de hachage ?

Pour créer une hashtable
```
/hmake -s <nom> N
```

L'attribue -s est facultatif, il sert à afficher dans la fenêtre de statut que la table a bien été créée. N est un entier donnant la taille de la table, mais pas en nombre d'items... Retenez juste que pour une table de 1000 éléments, une valeur de 100 est suffisante.
/hmake hashtable 10

Pour ajouter une clé et sa valeur à la table
```
/hadd <nom de la table> <clé> <valeur>
```

Aucun attribut n'est à retenir.

Pour supprimer une clé
```
/hdel <nom de la table> clé
```

Pour incrémenter ou décrémenter une valeur numérique
```
/hinc <nom> <clé> et /hdec <nom> <clé>
```

Pour sauvegarder une table de hachage
```
/hsave <nom de la table> <nom du fichier>
```

exemple
```
/hsave hashtable hashtable.txt
```

2 attributs utiles : -a pour écrire après les données déjà sauvegardée (si le fichier existait déjà avant) et -o pour ré écrire par dessus.

Pour charger une table précédemment sauvegardée
```
/hload <nom de la table> <nom du fichier>
```

Important : la table qui contiendra les données doit avoir été précédemment déclarée via hmake.

exemple
```
//hmake table 10 | hload table hashtable.txt
```

A noter qu'il n'y a aucune correspondance entre le nom de la table et le nom du fichier.

Enfin, pour supprimer une table de la mémoire vive, il faut utiliser :
```
/hfree <nom de la table>
```

Si l'on rajoute l'option -w, le nom de la table pourra comporter des jokers.

N'oubliez pas de vider la mémoire vive une fois que vou n'utilisez plus une table de hachage ! Si vous ne le faites pas, en plus de vous encombrer, vous ne pourrez plus créer une table du nom de celle que vous venez de fermer, ce qui peut être génant quand on a plusieurs alias se servant de la même table...

Quels sont les fonctions pour utiliser les tables de hachages ?

Passons aux fonctions permettant d'utiliser les hashtables.
Il n'y en a que trois: $hget ; $hmatch et $hfind, mais elles peuvent utilisée de bien des façons.

On commence par $hget() qui permet de retrouver la valeur d'une clé.

```
$hget(<nom de la table>,<clé>)
```

Retournera donc la valeur correspondant à la clé. A noter que <nom de la table> peut être remplacé par un entier, dans ce cas mIRC ira chercher dans la nième table.
Si vous voulez tester l'existence d'une table de hachage, la fonction $hget(<nom de la table>) retournera le nom de la table si elle existe, la chaîne vide ($null) sinon. Ca peut être utile.
Moins important, si l'on remplace <nom de la table> par un entier la fonction retourne le nom de la nième table.

Passons à $hmatch(). Elle permet de retrouver le nom d'une clé.

```
$hmatch(<nom de la table>,<chaîne avec jokers>,<n>)
```

Retourne la nième clé contenant la chaîne. Si n vaut 0, la fonction retourne le nombre de clé contenant la chaîne. Il vaut mieux encadrer la chaîne avec des * (comme dans on 1:*text*)

exemple
```
$hmatch(<nom de la table>,*,0)
```

Retournera le nombre de clés présentes dans la table, ce qui peut être très utile...

Les fonction $hget() et $hmatch() se combinent très bien !
```
$hget(<nom de la table>,$hmatch(<nom>,<chaîne>,<n>)
```

Permet donc de rechercher dans la table la valeur de la nième clé contenant chaîne.

La fonction $hmatch() permet aussi de rechercher dans les données plutôt que dans les clés (mais elle retournera toujours une clé) grâce à un .data après la parenthèse fermante.
Enfin, $hfind est équivalente à $hmatch() (en réalité dans la documentation on ne trouve pas la présentation de $hmatch() mais celle de $hfind(), mais je vous rassure, $hmatch() fonctionne très bien aussi)...

Retrouvez ces questions et de nombreuses autres sur la FAQ mIRC : Lien61

Liens

Lien1 : http://sznajderman.developpez.com/articles/java/covariance
Lien2 : http://www.eclipse.org/download/
Lien3 : http://www.easyeclipse.org/
Lien4 : http://www.myeclipseide.com/
Lien5 : http://www.obeo.fr/eclipse-download.php
Lien6 : http://eclipsediscovery.yoxos.com/discovery/rap
Lien7 : http://amateras.sourceforge.jp/
Lien8 : http://www.adobe.com/devnet/flex/ide.html
Lien9 : http://bcourtin.developpez.com/articles/eclipse/plugins
Lien10 : http://java.developpez.com/livres/?page=Anglais#L0072253606
Lien11 : http://java.developpez.com/livres/?page=Francais#L2212120389
Lien12 : http://java.developpez.com/livres/?page=Francais#L2100492195
Lien13 : http://java.developpez.com/faq/netbeans/
Lien14 : http://www.prendreuncafe.com/blog/post/2006/06/20/473-installer-le-framework-php-symfony-sur-ubuntu-dapper-drake
Lien15 : http://www.symfony-project.com/book/trunk/03-Running-symfony#Installing%20the%20symfony%20PEAR%20Package
Lien16 : http://www.glagla.org/blog/index.php/2007/03/11/163-installer-symfony-sous-mac-os-x-avec-mamp
Lien17 : http://trac.symfony-project.com/trac/wiki/symfonyOnWampEnFrancais
Lien18 : http://linux.developpez.com/faq/?filtre=NNNNONNNNNNNNNNNNNNNNN#sect5
Lien19 : http://andreiabohner.files.wordpress.com/2007/05/symfonycheat001fr.pdf
Lien20 : http://baptiste-wicht.developpez.com/tutoriel/conception/mvc/
Lien21 : http://tahe.developpez.com/web/php/mvc/
Lien22 : http://www.symfony-project.com/book/trunk/14- Generators#Administration
Lien23 : http://c-maneu.developpez.com/tutorial/web/php/symfony/intro/
Lien24 : http://lefortludovic.developpez.com/tutoriels/coldfusion/premiers-pas/
Lien25 : http://kiroukou.developpez.com/articles/sandy/v1.1/interpolateurs/
Lien26 : http://webman.developpez.com/articles/dotnet/texttospeech/
Lien27 : http://mbenzeghiba.developpez.com/tutoriels/dotnet/heritagevbnet
Lien28 : http://c.developpez.com/faq/cpp/?page=pointeurs#POINTEURS_raii
Lien29 : http://c.developpez.com/faq/cpp/?page=pointeurs#POINTEURS_intelligents
Lien30 : http://www.boost.org/libs/smart_ptr/smart_ptr.htm
Lien31 : http://c.developpez.com/faq/cpp/?page=divers#DIVERS_tr1
Lien32 : http://www.boost.org/doc/html/boost_tr1.html
Lien33 : http://arb.developpez.com/c++/boost/install/vc++/
Lien34 : http://www.boost.org/libs/smart_ptr/shared_ptr.htm#allocator_constructor
Lien35 : http://www.boost.org/libs/utility/utility.htm#Class_noncopyable
Lien36 : http://msdn2.microsoft.com/en-us/library/hsyx7kbz(VS.80).aspx
Lien37 : http://arb.developpez.com/c++/raii/shared_ptr/
Lien38 : http://gtk.developpez.com/livres/#L1590597931
Lien39 : http://schelabi.developpez.com/securitegranulaire/
Lien40 : http://fadace.developpez.com/oracle/downstreams/
Lien41 : http://sqlserver.developpez.com/livres/#L9782746035263
Lien42 : http://miles.developpez.com/tutoriels/hardware/config
Lien43 : http://baptiste-wicht.developpez.com/tutoriel/hardware/choix/boitier/
Lien44 : www.iis.net
Lien45 : http://webman.developpez.com/articles/windows/ssliis7/
Lien46 : http://hikage.developpez.com/linux/tutoriels/subversion/
Lien47 : http://fr.wikipedia.org/wiki/Tivoization
Lien48 : http://www.fsf.org/news/gplv3_launched
Lien49 : http://blog.developpez.com/?blog=139&p=3883
Lien50 : http://xgql.developpez.com/xgql/presentation/
Lien51 : http://xgql.sourceforge.net/
Lien52 : http://xgql.developpez.com/xgql/tutorial/
Lien53 : http://www.amazon.fr/Test-Driven-Development-Example-Kent-Beck/dp/0321146530/ref=sr_1_20/403-0522420-1320461?ie=UTF8&s=english-books&qid=1177932045&sr=8-20
Lien54 : http://smeric.developpez.com/java/astuces/tests/
Lien55 : http://jab.developpez.com/tutoriels/dotnet/nunit/
Lien56 : http://fr.wikipedia.org/wiki/Pentamino
Lien57 : http://www.microsoft.com/france/msdn/vstudio/express/vcsharp/telechargez.mspx
Lien58 : http://www.nunit.org/
Lien59 : http://bruno-orsier.developpez.com/tutoriels/TDD/pentaminos/
Lien60 : http://b-pino.developpez.com/tutoriaux/irc/dll-mirc/
Lien61 : http://irc.developpez.com/faq/mirc/