# Developpez Magazine

Edition de Février-Mars 2008.
Numéro 14.
Magazine en ligne gratuit.
Diffusion de copies conformes à l'original autorisée.
Réalisation : Baptiste Wicht et Alexandre Pottiez
Rédaction : la rédaction de Developpez
Contact : magazine@redaction-developpez.com

**Index**



## Editorial

En ce printemps pluvieux, la nouvelle édition du magazine Developpez vous propose une nouvelle sélection d'articles et autres ressources parues sur Developpez.com.

La rédaction

## Article Java


# Programmation par composants avec OSGi

*Mise en oeuvre de la programmation orientée composant et d'architectures orientées service en se fondant sur la technologie OSGi.*

par **Thierry Templier**
Page 2

## Article .NET


# Découverte de Microsoft Volta

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

par **Louis-Guillaume Morand**
Page 28

# Programmation par composant avec la technologie OSGi (1ère partie)

Cette série d'articles décrit la mise en oeuvre de la programmation orientée composant et d'architectures orientées service en se fondant sur la technologie OSGi. Nous y détaillerons les différents concepts de cette technologie afin de permettre sa prise en main.

## 1. Introduction

Ce premier article a pour objectif de fournir une introduction à la technologie OSGi (Open Service Gateway Initiative), technologie qui après avoir été longtemps utilisée dans le monde de l'embarqué est de plus en plus mise en oeuvre dans les applications classiques et serveurs.

Nous verrons dans un premier temps les différents concepts relatifs à OSGi et détaillerons à quelles problématiques cette technologie s'adresse tout en soulignant ses apports en terme d'architecture et de structuration des applications.

Nous rentrerons par la suite dans les aspects techniques d'OSGi afin de détailler ses caractéristiques et de voir concrètement comment mettre en oeuvre cette technologie au sein d'applications Java. Nous laisserons le soin à un prochain article de décrire la manière d'utiliser OSGi dans des applications Java EE tout en se fondant sur des bibliothèques et frameworks Java EE.

## 2. Programmation orientée composant et architecture orientée service avec OSGi

Avant d'entrer dans le coeur de la technologie, nous allons détailler les caractéristiques de la programmation orientée composant et des architectures orientée service afin d'identifier les problématiques qu'elles visent à résoudre. Nous verrons alors que la technologie OSGi adresse ces deux types de technologies.

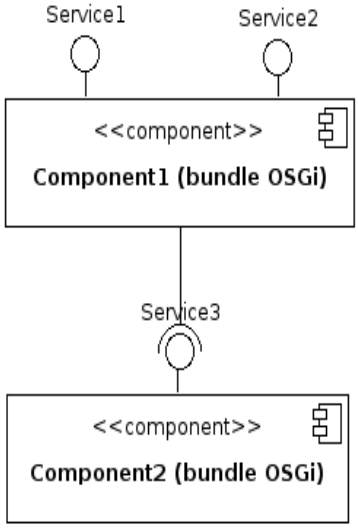
Tout d'abord, la programmation orientée composant [1] vise à adresser les limitations de la programmation orientée objet [2]. Cette dernière offre d'intéressants mécanismes afin de modulariser les traitements et les rendre réutilisables mais n'apporte aucun support afin de mettre en relation les classes. De plus, plus les traitements de l'application augmentent et se complexifient, plus ces limitations sont visibles et pénalisantes au niveau de la maintenabilité et de l'évolutivité de l'application.

Ainsi, cet aspect peut devenir très problématique lors de la mise en oeuvre de réseaux complexes d'objets. En effet, cela se traduit très souvent par des couplages forts entre objets, ce point nuisant énormément à la réutilisabilité des traitements. Chaque développeur a alors le choix des outils afin d'adresser au mieux au sein de ses applications ces aspects. L'approche la plus adaptée consiste en la mise en oeuvre du patron de conception injection de dépendance par l'intermédiaire de frameworks tels que Spring [3] utilisés conjointement avec la programmation par interface.

Nous retrouvons ce problème notamment au niveau des instanciations des objets, directement ou par l'intermédiaire de l'introspection. En effet, même avec la programmation par interface, la classe appelante est liée à la classe appelée. Les patrons de conception tels que les fabriques [4] et l'injection de dépendance [5] permettent d'adresser cette problématique en déléguant les traitements d'instanciation à une entité autonome (respectivement la fabrique et le conteneur).

En parallèle de la programmation orientée composant, les architectures orientées service peuvent être mises en oeuvre afin de mettre à disposition des traitements tout en diminuant les couplages entre les briques techniques. La SOA (Service Oriented Architecture) [6] n'implique pas nécessairement l'utilisation des services Web avec les technologies SOAP [7] et WSDL [8] et peut être mise en oeuvre au sein d'un même processus Java notamment avec des conteneurs légers (entités mettant en oeuvre l'injection de dépendances) tels que Spring [3] ou Hivemind [9]. Dans ce contexte, nous parlons de fournisseur de services et de consommateurs de services, les fournisseurs mettant à disposition des composants. La programmation orientée composant [1] et les architectures orientées service peuvent donc être mises en oeuvre de manière complémentaire afin de bénéficier des avantages des deux types de technologies.

La figure suivante illustre différents composants exposant des services par l'intermédiaire d'interfaces et reliés entre eux par des services:




Il est à noter que des composants OSGi peuvent être également reliés entre eux par l'intermédiaire des packages, packages qu'ils exportent et importent. Dans ce cas, les services ne sont donc pas l'unique manière de les relier.

Dans ce contexte, la technologie OSGi vise à adresser les différentes problématiques vues précédemment, problématiques récapitulées ci-dessous:

- Adresser les limitations de la programmation orientée objet;
- Permettre la gestion des applications complexes et de

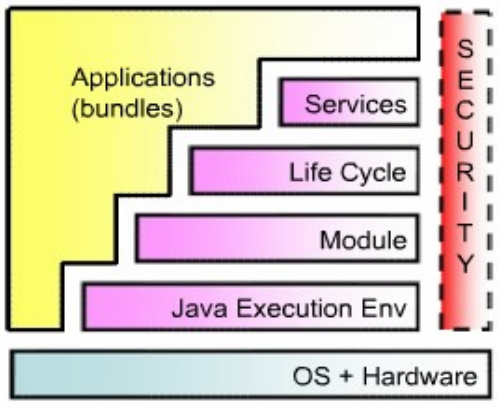
taille importante;
- Améliorer la qualité de service des applications en permettant une administration à chaud;
- Permettre la mise en oeuvre d'architectures orientées service légères.

Une des autres caractéristiques de la technologie OSGi est sa portabilité puisqu'elle peut être mise en oeuvre aussi bien dans des terminaux de manière embarquée que dans des applications classiques ou serveurs. Le premier aspect a été à la base d'OSGi, les seconds types d'application s'étant développés ces dernières années par l'intermédiaire d'outils tels qu'Eclipse [10]. Le fait qu'OSGi repose sur la technologie Java pour son exécution a été également un important facteur de portabilité.

## 3. Architecture OSGi

Dans cette section, nous allons décrire les caractéristiques des conteneurs et composants OSGi. Le composant correspond à l'entité centrale de la technologie et désignée par le terme bundle avec la terminologie de cette dernière. Nous verrons également les spécificités du cycle de vie des bundles, la manière de les gérer au niveau du conteneur ainsi que la façon de les configurer par l'intermédiaire de leur descripteur de déploiement.

La figure suivante décrit les différentes briques de l'architecture de la technologie OSGi, briques que nous allons détaillées conceptuellement tout au long de cette section puis techniquement dans le reste de l'article:




La couche Module adresse la gestion des bundles, aussi bien au niveau du chargement des classes (classloading) que de la gestion de leurs visibilités et de leurs versions. La couche Cycle de vie (life cycle) prend quant à elle en charge les états des bundles supportés par le conteneur ainsi que les API correspondantes. Pour finir, la couche Services offre la possibilité de mettre à disposition des services au sein d'une même machine virtuelle tout en masquant leurs implémentations à leurs utilisateurs. Les applications OSGi peuvent se fonder et tirer parti de ces différentes couches afin de mettre en oeuvre des composants et des services.

### 3.1. Gestion des composants OSGi

La couche module permet la mise en oeuvre des bundles dans le conteneur. Elle a la responsabilité de la gestion des différents chargeurs de classes et des versions des dépendances.

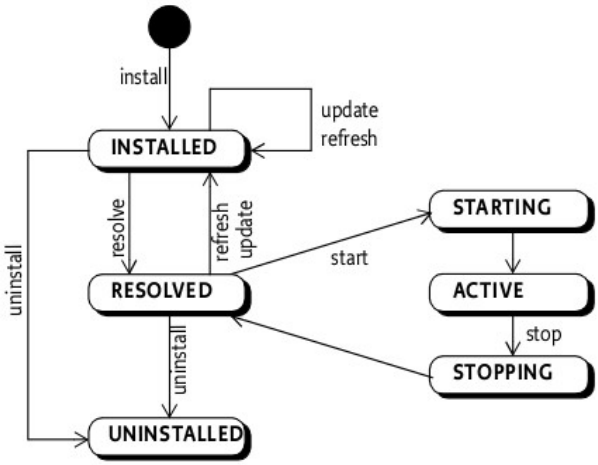
En effet, une des caractéristiques de la technologie consiste en l'isolation des classloaders des composants. En effet, chaque composant ou bundle OSGi dispose d'un classloader indépendant pour ses traitements. Cet aspect permet par exemple la mise en oeuvre dans deux bundles différents d'une même bibliothèque avec deux versions différentes et incompatibles.

Cet aspect offre la possibilité à OSGi de complètement maîtriser les classes et packages du composant visibles depuis l'extérieur et visibles par les autres composants. Par défaut, la stratégie est la plus restrictive possible, à savoir que rien n'est visible. Par contre, lors de la mise à disposition d'un package d'un composant, un numéro de version spécifique peut être spécifié. Ainsi, deux versions de packages peuvent coexister dans le conteneur OSGi sans aucun problème.

### 3.2. Cycle de vie des bundles

Comme dans tout conteneur, les éléments contenus dans ce dernier possèdent un cycle de vie bien particulier puisque le conteneur OSGi a la responsabilité de les gérer. Nous pouvons distinguer deux grandes parties dans les différents états supportés:
- **Non opérationnel**. Les états correspondent à la présence du bundle dans le conteneur, mais ce dernier n'est pas utilisable par les autres bundles pour leurs traitements;
- **Opérationnel**. Les états de ce type correspondent aux moments où le bundle est utilisable ou en phase de l'être ou de ne plus l'être.

La figure suivante illustre les différents états de gestion des bundles par le conteneur ainsi que leurs enchaînements possibles:




Nous verrons par la suite que le conteneur OSGi offre une API standardisée afin de gérer le cycle de vie des composants. Veuillez vous reporter à la section 5.2 BundleContext pour plus de précisions. De plus, ces différents états sont mis en oeuvre par le bundle manager du conteneur qui a la responsabilité de gérer la plateforme.

Le tableau suivant récapitule les différents états supportés par le bundle manager des conteneurs OSGi ainsi que leurs principales caractéristiques:

| Etat | Descriptif |
|---|---|
| Installé (installed) | Etat dans lequel se trouve un bundle juste après avoir été installé, la résolution des dépendances n'ayant pas encore été réalisée. |
| Résolu (resolved) | Etat dans lequel se trouve un bundle après avoir été installé, la résolution des dépendances ayant juste été réalisée. |

| | |
|---|---|
| En train de démarrer (starting) | Etat dans lequel se trouve un bundle lorsqu'il est en train d'être démarré. Cet état correspond à un état transitoire entre les événements Résolu et Actif. |
| Actif (active) | Etat dans lequel se trouve un bundle lorsqu'il a été démarré avec succès. Le bundle ainsi que les services qu'il expose sont disponibles pour les autres bundles. |
| En train de s'arrêter (stopping) | Etat dans lequel se trouve un bundle lorsqu'il est en train d'être arrêté. Cet état correspond à un état transitoire entre les événements Actif et Résolu. |
| Désinstallé (uninstalled) | Etat dans lequel se trouve un bundle une fois qu'il a été désinstallé. |

Nous pouvons noter qu'une entité peut être associée au cycle de vie des bundles et correspond à l'entité d'activation, entité appelée lors du démarrage et de l'arrêt d'un bundle. Cette entité peut être mise en oeuvre par l'intermédiaire de l'interface BundleActivator et configurée avec l'en-tête Bundle-Activtator. Nous reviendrons plus en détail sur ce mécanisme dans la section 5.1 BundleActivator.

Nous pouvons noter que la plupart des conteneurs OSGi fournissent des outils afin de gérer le cycle de vie des bundles et de visualiser leurs états. Dans cette optique, le conteneur OSGi
Felix [11] offre une console d'administration en ligne de commande possédant les commandes suivantes:

| Commande | Descriptif |
|---|---|
| install | Installation de bundle(s). |
| ps | Affichage de la liste des bundles installés. |
| refresh | Rafraichissment des packages des bundles. |
| resolve | Tentative de résolution des bundles spécifiés. |
| services | Affichage de la liste des services enregistrés ou utilisés. |
| start | Démarrage de bundle(s). |
| stop | Arrêt de bundle(s). |
| uninstall | Désinstallation de bundle(s). |
| update | Mise à jour d'un bundle. |

Avec la console de cet outil, le conteneur peut être complètement administré à chaud en ligne de commandes. Le scénario suivant peut être mis en oeuvre afin d'installer un bundle, de le démarrer, puis l'arrêter avec les commandes respectives install, start et stop. Le code suivant illustre concrètement l'enchaînement de ces différentes commandes:

```
-> ps
START LEVEL 1
   ID   State         Level  Name
[   0] [Active     ] [    0] System Bundle (1.0.0)
[   1] [Active     ] [    1] Apache Felix Shell Service
(1.0.0)
[   2] [Active     ] [    1] Apache Felix Shell TUI
(1.0.0)
-> install
/home/templth/developpez/OSGi/simplebundle.jar
Bundle ID: 3
-> ps
START LEVEL 1
   ID   State         Level  Name
[   0] [Active     ] [    0] System Bundle (1.0.0)
[   1] [Active     ] [    1] Apache Felix Shell
Service (1.0.0)
[   2] [Active     ] [    1] Apache Felix Shell TUI
(1.0.0)
[   3] [Installed  ] [    1] Simple Bundle (1.0.1)
-> start 3
-> ps
START LEVEL 1
   ID   State         Level  Name
[   0] [Active     ] [    0] System Bundle (1.0.0)
[   1] [Active     ] [    1] Apache Felix Shell
Service (1.0.0)
[   2] [Active     ] [    1] Apache Felix Shell TUI
(1.0.0)
[   3] [Active     ] [    1] Simple Bundle (1.0.1)
-> stop 3
-> ps
START LEVEL 1
   ID   State         Level  Name
[   0] [Active     ] [    0] System Bundle (1.0.0)
[   1] [Active     ] [    1] Apache Felix Shell
Service (1.0.0)
[   2] [Active     ] [    1] Apache Felix Shell TUI
(1.0.0)
[   3] [Resolved   ] [    1] Simple Bundle (1.0.1)
```

## 3.3. Services

La couche Services offre la possibilité de mettre à disposition certains traitements des composants par l'intermédiaire de services. Nous désignons par le terme service des traitements structurés de la manière suivante. Il s'agit d'une entité dont le contrat est clairement défini par l'intermédiaire d'une interface Java. Seule cette interface est connue par les consommateurs du service. La ou les implémentations du service doivent rester internes aux composants et implémenter la précédente interface.

Le mécanisme de gestion des services est complètement dynamique. En effet, dès qu'un service est enregistré, il est automatiquement utilisable par tous les composants disponibles dans le conteneur OSGi. Comme nous le verrons par la suite, une API est disponible afin d'enregistrer des implémentations de services. De plus, par convention, il convient de faire correspondre le nom du service au nom de l'interface implémentée.

Dans le cadre de la technologie OSGi, l'architecture orientée services a la caractéristique d'être très "légère" et de pouvoir fonctionner de manière autonome dans un seul processus Java. Elle correspond à un modèle de services dans une unique machine virtuelle (in-VM).

Nous verrons dans la section 5.1 comment mettre en oeuvre concrètement des services dans le cadre de la technologie OSGi.

## 3.4. Conclusion

L'architecture de la technologie OSGi permet de mettre en oeuvre des composants et de mettre à disposition des services applicatifs. Elle offre également un cycle de déploiement de services très rapides par l'intermédiaire du cycle de vie des bundles tout en offrant une meilleure structuration des fondations des applicatifs.

Nous pouvons remarquer que la sécurité est prévue en standard puisqu'elle fait partie intégrante de la spécification OSGi. Elle offre la possibilité de spécifier des permissions d'accès aux entités gérées par le conteneur. Une de ses principales caractéristiques consiste dans le fait que sa configuration est complètement dynamique et ne nécessite aucun redémarrage après une modification.

Rentrons maintenant dans le détail technique de la technologie OSGi et voyons concrètement comment la mettre en oeuvre.

## 4. Bundle OSGi

Dans cette section, nous allons détailler les différentes caractéristiques de l'entité centrale d'OSGi, le composant ou bundle. Nous verrons comment le mettre en oeuvre par l'intermédiaire d'un simple fichier JAR de Java.

### 4.1. Caractéristiques d'un bundle

Avec la technologie OSGi, le concept de composant est mis en oeuvre par l'intermédiaire des bundles, un bundle correspondant à un composant. Ces derniers permettent de mettre en oeuvre les différents concepts des composants et ce sont eux qui sont déployés dans les conteneurs OSGi.

Avec OSGi, un composant (bundle) est simplement stocké dans un fichier JAR de Java. Les informations de déploiement sont spécifiées par l'intermédiaire du fichier standard MANIFEST.MF présent dans le répertoire META-INF. En effet, OSGi reprend ce fichier en y ajoutant différents en-têtes afin de configurer le composant.

Il n'est pas nécessaire d'ajouter d'autres éléments au fichier JAR si ce n'est les classes bien sûr et les éventuelles bibliothèques utilisées, uniquement les bibliothèques n'étant pas définies en tant que dépendances. Nous pouvons constater que cet aspect favorise de manière importante la simplicité de mise en oeuvre de composants dans des conteneurs de ce type. Les conteneurs OSGi sont de ce fait très légers. N'oublions pas qu'OSGi était à l'origine utilisé dans des systèmes embarqués donc très soucieux de la consommation mémoire.

Les conteneurs possèdent diverses caractéristiques quant à la gestion des composants. Tout d'abord, ils ont la particularité de permettre la gestion des dépendances entre composants. En effet, comme nous l'avons vu précédemment, un composant s'appuie la plupart du temps sur d'autres afin de réaliser ses traitements et de fonctionner correctement. Pour ce faire, OSGi offre la possibilité de rendre visible dans le conteneur les packages de composants de manière très fine. La gestion des versions des dépendances est également supportée.

### 4.2. en-têtes OSGi du fichier MANIFEST.MF

Comme nous l'avons précisé dans la précédente section, le fichier MANIFEST.MF correspond au descripteur de déploiement du bundle. Ce fichier est lu par le conteneur OSGi afin de configurer le bundle.

La spécification OSGi définit un ensemble d'en-têtes standards pour un composant, en-têtes utilisables dans le fichier MANIFEST.MF et dont la liste des principales est récapitulée dans le tableau suivant:

| En-tête | Descriptif |
|---|---|
| Bundle-ManifestVersion | Correspond à la version de fichier MANIFEST du bundle |
| Bundle-SymbolicName | Spécifie l'identifiant symbolique du bundle |
| Bundle-Name | Spécifie le nom du bundle |
| Bundle-Version | Spécifie la version du bundle |
| Bundle-DocURL | Permet de préciser l'adresse de la documentation du bundle. |
| Bundle-Category | Spécifie la catégorie du bundle. |
| Import-Package | Spécifie les noms et les versions des packages utilisés par le bundle. |
| Export-Package | Spécifie les noms et les versions des packages mis à disposition par le bundle. |
| DynamicImport-Package | Spécifie les noms et les versions des packages utilisés par le bundle. Cet en-tête se différencie de Import-Package par le fait qu'il ne soit pas nécessaire que les dépendances soient présentes au démarrage du bundle. Il suffit qu'elles le soient au moment de l'exécution. |
| Bundle-NativeCode | Spécifie la liste des bibliothèques natives présentes dans le bundle. |
| Require-Bundle | Spécifie les identifiants symboliques des bundles nécessaires au bon fonctionnement du bundle. |
| Bundle-Activator | Spécifie le nom de la classe dont les traitements sont exécutés lors du démarrage et de l'arrêt du bundle. Cette classe doit être présente dans le bundle et implémenter l'interface BundleActivator. |
| Bundle-Classpath | Spécifie le classpath du bundle. Par défaut, la valeur implicite correspond à . et il faut veiller à ne pas l'oublier lorsque des bibliothèques sont spécifiées explicitement. |

Le code suivant illustre la configuration d'un composant OSGi simple dans le fichier MANIFEST.MF. Ce bundle est identifié par simple.bundle, possède le nom "Simple Bundle", exporte le package org.developpez.osgi.service et utilise l'entité d'activation org.developpez.osgi.SimpleActivator.

```
Manifest-Version = 1
Bundle-Activator = org.developpez.osgi.SimpleActivator
Bundle-SymbolicName = simple-bundle
Bundle-Name = Simple Bundle
Bundle-Description = Simple Bundle.
Import-Package = org.osgi.framework;version=1.3
Export-Package = org.developpez.osgi.services
Bundle-Version = 1.0.1
Bundle-License =
http://www.apache.org/licenses/LICENSE-2.0.txt
```

## 5. Interactions entre bundles

Comme nous l'évoquions dans la section précédente, un conteneur OSGi offre la possibilité de gérer la visibilité des ressources en composant. Par défaut, un composant est complètement opaque depuis l'extérieur et rien n'est accessible. Cet aspect implique de préciser toutes les dépendances utilisées par un composant et tous les packages mis à disposition.

La spécification OSGi offre la possibilité de jouer sur cet aspect par l'intermédiaire d'en-têtes dans le fichier MANIFEST.MF, fichier contenant les données relatives au déploiement et décrit dans la précédente section. Détaillons maintenant les spécificités de ces en-têtes et la manière de les mettre en oeuvre.

Notons qu'il existe une manière complémentaire de faire interagir les bundles entre eux par l'intermédiaire des services. Nous décrirons cet aspect dans une prochaine section.

### 5.1. Mise à disposition de packages

Afin de mettre à des dispositions des packages (ou exporter) d'un bundle, l'en-tête Export-Package doit être mise en oeuvre dans le fichier MANIFEST.MF du composant. Cet en-tête contiendra la liste des packages pour lesquels l'accès est possible.

Il est à noter que la spécification d'un package dans ce cadre autorise l'accès à toutes les classes du package tout en gardant inaccessible les classes contenues dans les sous packages du package.

Lors de l'exportation d'un package, OSGi offre la possibilité d'ajouter des informations complémentaires telles que la version et dont la liste est récapitulée dans le tableau ci-dessous:

| Paramètre | Descriptif |
|---|---|
| uses | Spécifie la liste des packages utilisés par le package exporté. |
| mandatory | Spécifie une liste de paramètres obligatoires à spécifier lors de l'importation du package exporté. |
| include | Spécifie une liste de packages devant être visibles par le bundle important le package. |
| exclude | Spécifie une liste de packages devant être invisibles par le bundle important le package. Spécifie la version avec laquelle est mis à disposition le package. |
| version | Spécifie la version avec laquelle est mis à disposition le package. |

La syntaxe complète d'une valeur pour la propriété Export-Package est donc la suivante:

```
Export-Package: nom-package;parametre:=valeur,nom-package;parametre:=valeur,...
```

Notons que, si la valeur d'un paramètre est une liste, les éléments doivent être séparés par des virgules et la valeur globale doit être entourée par des guillemets.

Le code suivant correspond à un extrait de la valeur de l'en-tête Export-Package du fichier MANIFEST.MF d'un bundle OSGi pour Hibernate 3 [12]. Nous remarquons la présence des paramètres uses et version pour le package org.hibernate.stat.

```
Export-Package:
org.hibernate.stat;uses:="org.hibernate.util,org.hibernate.cache,
 org.apache.commons.logging,org.hibernate.engine";version=3.1.3,
 org.hibernate.cache;uses:="org.hibernate.util,net.sf.swarmcache,
 org.hibernate.cfg,org.hibernate,net.sf.ehcache,org.hibernate.impl,...
```

### 5.2. Importation de packages

La technologie OSGi offre la possibilité de spécifier les packages utilisés afin qu'ils soient visibles dans le bundle. Une fois spécifiées, les classes sont utilisables dans les classes implémentées par le bundle. Cet aspect se configure dans le fichier MANIFEST.MF par l'intermédiaire des en-têtes Import-Package et DynamicImport-Package. Détaillons tout d'abord l'usage du premier en-tête.

Lors de l'importation d'un package, OSGi offre la possibilité d'ajouter des informations complémentaires telles que la version et le mode de résolution dont la liste simplifiée est récapitulée dans le tableau ci-dessous:

| Paramètre | Descriptif |
|---|---|
| resolution | Spécifie le type de résolution du package. La valeur par défaut est mandatory. |
| version | Spécifie la version du package à utiliser. Une plage de versions peut être spécifiée. |

La syntaxe complète d'une valeur pour la propriété Import-Package est donc la suivante:

```
Import-Package: nom-package;parametre:=valeur,nom-package;parametre:=valeur,...
```

Notons que, si la valeur d'un paramètre est une liste, les éléments doivent être séparés par des virgules et la valeur globale doit être entourée par des guillemets.

Le code suivant correspond à un extrait de la valeur de l'en-tête Import-Package du fichier MANIFEST.MF du bundle OSGi de Spring AOP 2.0.5. Nous remarquons la présence des paramètres resolution pour le package com.jamonapi et version pour le package org.springframework.aop.

```
Import-Package:
com.jamonapi;resolution:=optional,net.sf.cglib.core;resolution:=optional,
 net.sf.cglib.proxy;resolution:=optional,net.sf.cglib.transform.impl;resolution:=optional,
 ...
 org.springframework.aop;version=2.0.5,org.springframework.aop.aspectj;version=2.0.5,...
```

Dans le cas de l'importation de packages, l'en-tête DynamicImport-Package est également disponible. Il se différencie de la précédente par le fait que la résolution des dépendances spécifiées à ce niveau n'est réalisée qu'au moment du chargement des classes et non lors du passage du bundle de l'état Installé à Résolu.

L'en-tête fonctionne sinon de la même manière que l'en-tête Import-Package, si ce n'est qu'il ne supporte logiquement pas le paramètre resolution.

## 6. Principales API OSGi

Par cette section, nous allons introduire deux interfaces des APIs de la technologie OSGi relatives respectivement à l'entité d'activation et au contexte des bundles. Elles permettent notamment d'initialiser et de finaliser les ressources des bundles et d'interagir avec le conteneur OSGi.

L'entité relative au contexte OSGi nous sera utile dans la section relative aux services puisque cette interface met à disposition des méthodes afin de manipuler et d'avoir accès aux services présents dans le conteneur OSGi.

### 6.1. BundleActivator - Entité d'activation

La technologie OSGi offre la possibilité de spécifier une entité appelée lors de l'activation (état Démarrage) et de la désactivation (état En cours d'arrêt) d'un composant, c'est-à-dire lorsque ce dernier est démarré ou arrêté. La classe implémentant cette entité doit nécessairement posséder l'interface BundleActivator et être configurée par l'intermédiaire de l'en-tête Bundle-Activator dans le fichier MANIFEST.MF.

L'interface BundleActivator définit la structure d'une classe d'activation en définissant les signatures des méthodes appelées lors des différents événements précédemment cités:

- Méthode **start**, appelée lors le démarrage du composant (état Démarrage);
- Méthode **stop**, appelée quant à elle lors de l'arrêt du composant (état En cours d'arrêt).

Ces deux méthodes prennent en paramètre le contexte OSGi du composant par l'intermédiaire de l'interface BundleContext, interface permettant d'interagir aussi bien avec le conteneur qu'avec le composant. Le contenu de l'interface BundleActivator est décrit ci-dessous:

```
public interface BundleActivator {
    void start(BundleContext context);
    void stop(BundleContext context);
}
```

Le code suivant illustre la mise en oeuvre d'une entité d'activation pour un bundle par l'intermédiaire d'une implémentation simple appelée ici SimpleActivator:

```
public class SimpleActivator implements BundleActivator
{
    private ServiceRegistration serviceRegistration;

    public void start(BundleContext context) {
        // Enregistrement d'un service
        SimpleService service  = new
SimpleServiceImpl();
        this.serviceRegistration =
context.registerService(
                    SimpleService.class.getName(),
service, null);
    }

    public void stop(BundleContext context) {
        // Désenregistrement d'un service
        if( this.serviceRegistration!=null ) {
            this.serviceRegistration.unregister();
        }
    }
}
```

```
Le code suivant décrit la configuration de cette entité
par l'intermédiaire de l'en-tête Bundle-Activator dans
le fichier MANIFEST.MF du bundle:
Manifest-Version = 1
Bundle-Activator = org.developpez.OSGi.SimpleActivator
Bundle-SymbolicName = simple-bundle
Bundle-Name = Simple Bundle
Bundle-Description = Simple Bundle.
Import-package = org.OSGi.framework;version=1.3
Bundle-Version = 1.0.1
Bundle-License =
http://www.apache.org/licenses/LICENSE-2.0.txt
```

### 6.2. BundleContext - Contexte de composant

Nous avons vu dans la précédente section que l'entité d'activation s'appuie sur le contexte du composant. Détaillons maintenant les spécificités de l'interface BundleContext, interface relative au contexte du composant. Cette dernière permet d'interagir aussi bien au niveau du conteneur OSGi lui-même que du bundle dans lequel il est utilisé. Elle offre la possibilité de réaliser les différentes opérations suivantes:

| Opération | Descriptif |
|---|---|
| Manipulation des bundles | Permet de récupérer les instances de bundles du conteneur et d'installer de nouvelles instances. |
| Manipulation des services | Permet de récupérer les instances de services du conteneur et d'installer de nouveaux. Nous détaillerons cet aspect dans la section suivante. |
| Observateurs | Permet de spécifier et de supprimer des observateurs à différents niveaux. |

Le contenu de l'interface BundleContext est décrit dans le code ci-dessous:

```
public interface BundleContext {
    void addBundleListener(BundleListener listener);
    void addFrameworkListener(FrameworkListener
listener);
    void addServiceListener(ServiceListener listener);
    void addServiceListener(ServiceListener listener,
                            Java.lang.String filter);
    Filter createFilter(Java.lang.String filter);
    ServiceReference[]
getAllServiceReferences(Java.lang.String clazz,
                                                Java.la
ng.String filter);
    Bundle getBundle();
    Bundle getBundle(long id);
    Bundle[] getBundles();
    Java.io.File getDataFile(Java.lang.String
filename);
    Java.lang.String getProperty(Java.lang.String
key);
    Java.lang.Object getService(ServiceReference
reference);
    ServiceReference
getServiceReference(Java.lang.String clazz);
    ServiceReference[]
getServiceReferences(Java.lang.String clazz,
                                                Java.lang.
String filter);
    Bundle installBundle(Java.lang.String location);
    Bundle installBundle(Java.lang.String location,
                         Java.io.InputStream input);
    ServiceRegistration
registerService(Java.lang.String[] clazzes,
                                        Java.lang.Obje
ct service,
                                        Java.util.Dict
ionary properties);
```

```
    ServiceRegistration registerService(Java.lang.String
clazz,
                                        Java.lang.Object
service,
                                        Java.util.Dictio
nary properties);
    void removeBundleListener(BundleListener listener);
    void removeFrameworkListener(FrameworkListener
listener);
    void removeServiceListener(ServiceListener
listener);
    boolean ungetService(ServiceReference reference);
}
```

Détaillons maintenant les spécificités relatives aux bundles et aux observateurs. Cette interface offre tout d'abord la possibilité de gérer et récupérer les instances des bundles présents dans le conteneur. Tout d'abord, l'installation de bundles se réalise par l'intermédiaire des méthodes installBundle, méthode prenant en paramètre le chemin du fichier JAR du bundle. Les méthodes getBundle et getBundles retournent quant à elle une ou plusieurs instances de bundles. Sont supportées les récupérations du bundle courant, d'un bundle en se fondant sur son identifiant dans le conteneur ou de tous les bundles du conteneur.

Le code suivant illustre la mise en oeuvre de ces méthodes dans une entité ayant accès au contexte OSGi:

```
// Installation d'un nouveau bundle
Bundle bundleInstalle = contexte.installBundle(
            "file:///home/templth/developpez/OSGi/simple
bundle.jar");
long idBundle = bundleInstalle.getBundleId();

// Récupération d'une instance d'un bundle présent dans
le conteneur
Bundle bundle = contexte.getBundle(idBundle);
int etatBundle = bundle.getState();
```

Comme vous pouvez le constater, le code précédent repose sur l'interface Bundle correspondant à l'entité relative à un composant OSGi. Cette dernière permet de récupérer des informations sur le bundle mais également de gérer son état. Le code suivant décrit le contenu de l'interface Bundle:

```
public interface Bundle {
    public static final int UNINSTALLED = 0x00000001;
    public static final int INSTALLED = 0x00000002;
    public static final int RESOLVED = 0x00000004;
    public static final int STARTING = 0x00000008;
    public static final int STOPPING = 0x00000010;
    public static final int ACTIVE = 0x00000020;

    Enumeration findEntries(String path, String
filePattern, boolean recurse)
    BundleContext getBundleContext()
    long getBundleId()
    URL getEntry(String path)
    Enumeration getEntryPaths(String path)
    Dictionary getHeaders()
    Dictionary getHeaders(String locale)
    long getLastModified()
    String getLocation()
    ServiceReference[] getRegisteredServices()
    URL getResource(String name)
    Enumeration getResources(String name)
    ServiceReference[] getServicesInUse()
    int getState()
    String getSymbolicName()
    boolean hasPermission(Object permission)
    Class loadClass(String name)
    void start()
    void start(int options)
    void stop()
    void stop(int options)
    void uninstall()
    void update()
    void update(InputStream in)
}
```

Il est ainsi possible par la programmation, après l'installation d'un bundle, de réaliser le démarrage du bundle et de récupérer les différents services mis à disposition par le bundle puis de l'arrêter, comme l'illustre le code suivant:

```
// Installation d'un nouveau bundle
Bundle bundleInstalle = contexte.installBundle(
            "file:///home/templth/developpez/OSGi/simp
lebundle.jar");

// Démarrage du bundle (état installed vers active)
bundle.start();

// Récupération des services mis à disposition
ServiceReference[] serviceReferences =
bundle.getRegisteredServices();

// Arrêt du bundle (état active vers resolved)
bundle.stop();
```

Pour finir, le contexte offre la possibilité de spécifier des observateurs d'événements sur le conteneur OSGi lui-même, sur les bundles et sur les services. Cet aspect est couramment utilisé dans les bonnes pratiques de mise en oeuvre de la technologie OSGi, notamment par l'intermédiaire du patron de conception Extender [13] au niveau des bundles.

Le principe consiste en la mise en oeuvre d'un observateur utilisant les données des bundles afin de réaliser des traitements. Ce patron permet de mieux gérer les états des bundles lors de traitements généraux, traitements pouvant être réalisés de manière paresseuse. Cette approche est mise en oeuvre dans des outils tels que Declarative Services [14], iPOJO [15], Spring Dynamic Modules [16].

Un observateur de bundles peut être mis en oeuvre par l'intermédiaire des interfaces BundleListener et SynchronousBundleListener, respectivement déclenchées de manière asynchrone et synchrone. Les deux interfaces mettent en oeuvre la méthode bundleChanged afin de notifier un changement pour un bundle. Cette méthode prend en paramètre un objet de type BundleEvent, objet permettant d'avoir accès au bundle impacté et à l'événement déclencheur dans le cycle vie. Le code suivant illustre le contenu de l'interface BundleListener:

```
public interface BundleListener extends EventListener
{
    void bundleChanged(BundleEvent event);
}
```

La mise en oeuvre d'un observateur de bundles se réalise de la manière suivante:

```
// Enregistrement de l'observateur
BundleListener observateur = new BundleListener() {
    public void bundleChanged(BundleEvent evenement) {
        // Récupération du bundle impacté
        Bundle bundle = evenement.getBundle();
```

```
        int etat = evenement.getType();
        (...)
    }
};

context.addBundleListener(observateur);
```

## 7. Mise en oeuvre de services

Comme nous l'avons évoqué précédemment, la technologie OSGi offre la possibilité de mettre des services au niveau des composants. Elle permet de bien séparer le contrat du service (interface) de la ou des implémentations fournies par les composants. Les consommateurs du service n'ont connaissance que de son interface. De plus, les services mis en oeuvre avec OSGi n'ont aucune adhérence avec les API de conteneurs et consistent en de simples POJOs [17].

La technologie OSGi, par l'intermédiaire de l'interface BundleContext, offre la possibilité à un composant de mettre à disposition des services par l'intermédiaire de la méthode registerService. Le premier paramètre de cette méthode correspond au nom du service, nom visible au niveau du conteneur. Un usage courant consiste à spécifier le nom de l'interface du service comme nom du service.

Le code suivant décrit la manière d'enregistrer un service dans un conteneur OSGi par l'intermédiaire du contexte OSGi:

```
SimpleService service  = new SimpleServiceImpl();
ServiceRegistration serviceRegistration =
context.registerService(
                              SimpleService.class.getNa
me(), service, null);
```

Le désenregistrement d'un service OSGi se réalise par l'intermédiaire de la méthode unregister de l'instance de ServiceRegistration renvoyée précédemment par la méthode register. Une bonne pratique consiste donc à le garder en variable d'instance (de l'activateur par exemple). Le code suivant illustre la mise en oeuvre d'un désenregistrement de service:

```
ServiceRegistration serviceRegistration = (...)
serviceRegistration.unregister();
```

Différentes méthodes sont également fournies afin d'avoir accès aux services afin de les utiliser. La récupération d'une instance de service se réalise en deux étapes. La première consiste en la récupération d'une instance de l'interface ServiceReference pour le service par l'intermédiaire de la méthode getServiceReference ou getServiceReferences. L'utilisation de la méthode getService avec en paramètre l'instance précédente permet d'avoir accès à l'instance du service et de l'utiliser.

Le code suivant décrit la manière d'avoir accès à une instance d'un service à partir de son nom et par l'intermédiaire du contexte OSGi:

```
// Récupération de la référence du service
ServiceReference reference =
context.getServiceReference(
                                        SimpleService.
class.getName());

// Récupération de l'instance du service
SimpleService service = context.getService(reference);
(...)
```

Il est à noter que, dans le cas d'une utilisation d'un service en dehors du conteneur, le transtypage dans le type du service ne fonctionnera pas. En effet, les classes et interfaces gérées par le conteneur ne sont visibles et utilisables qu'en son sein. Un service OSGi reste néanmoins utilisable en se fondant sur l'introspection.

Nous pouvons également noter la présence d'une méthode ungetService au niveau du contexte afin de libérer l'instance du service référencée par l'instance de ServiceReference correspondante.

## 8. Conclusion

Dans ce premier article, nous avons introduit la technologie OSGi et les différentes problématiques qu'elle vise à adresser. Nous avons également décrit les caractéristiques de la technologie et comment mettre en oeuvre des composants OSGi, composants désignés par le terme bundle dans la terminologie d'OSGi. Nous avons ainsi abordés les différentes couches structurant les conteneurs de ce type, à savoir les couches Module, Cycle de vie et Service.

Dans le prochain article, nous verrons comment mettre concrètement en oeuvre des composants OSGi dans un conteneur embarqué dans un processus Java. Pour ce faire, nous utiliserons le conteneur OSGi libre Felix. Nous verrons également comment tirer partie de l'environnement Eclipse afin de développer des bundles OSGi.

Retrouvez la suite de l'article de Thierry Templier avec ses références en ligne : Lien1

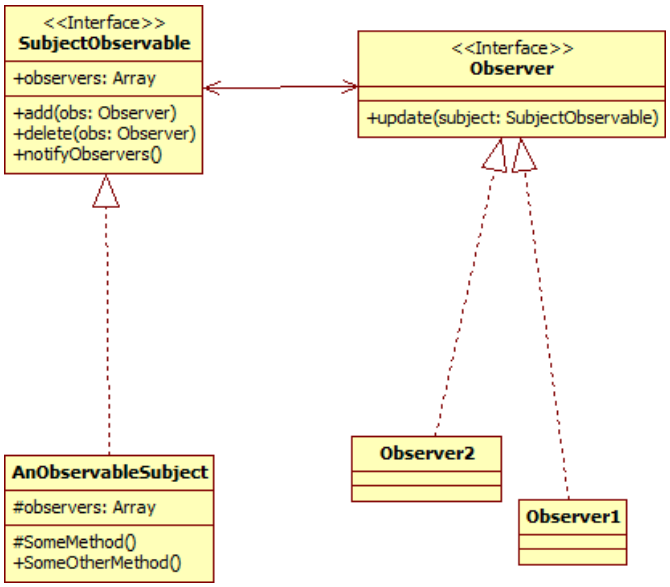
# POO PHP5 : Design Pattern observateur aidé de la Standard PHP Library (SPL)

Le design pattern observateur est un classique du GOF, il participe au découplage et à la réduction des dépendances.

En général, 2 interfaces sont utilisées, on peut aussi manipuler des classes abstraites. Nous allons ici montrer un exemple complet de son utilisation et nous allons nous aider de la puissante librairie objet interne de PHP5 : la SPL.

## 1. Introduction au design pattern observateur

Le design pattern observateur autorise un sujet observable, à enregistrer des observateurs. Sur une certaine action, il va notifier ses observateurs, qui vont donc être tenus au courant de ses agissements et changements d'état.
Un sujet pourra donc contenir plusieurs observateurs qui l'écouteront et réagiront à certains de ses évènements.
Le sujet observable ne s'occupe pas de la manière dont ses observateurs vont réagir, l'application gagne donc en découplage et en cohésion : il ne fait que notifier les observateurs, qui vont agir en conséquence, selon leur type et l'état du sujet transmis.
Ce design pattern est donc flexible et extensible, même si on pourra lui trouver des inconvénients.
En général, il fait appel à 2 interfaces, plus rarement à des classes abstraites. Le couplage se situe au niveau des interfaces, et non plus de leur implémentation.
On l'appelle "Observeur-Observable", ou encore "observateurs-sujet" ("Publish/Subscribe").




La théorie la plus généraliste est donc exprimée via le diagramme de classes ci-dessus. Le sujet et les observateurs sont fortement couplés, mais le design pattern représenté par les interfaces permet de centraliser ce couplage au niveau des interfaces. Je n'ai pas représenté l'implémentation concrète des interfaces (les méthodes dans les classes sujet et observateurs).
Simplement, la méthode notifyObservers() aura un algorithme ressemblant à "pour tous les observateurs enregistrés : execute update($this) dessus".
De cette manière, le sujet observable n'a que faire de la quantité d'observateurs qui le regardent, il ne fait que leur donner un ordre de mise à jour en leur passant lui-même ($this). Après : chacun des observateurs fait ce qu'il désire du sujet.

## 2. Exemple : un gestionnaire d'erreur PHP

Notre exemple est lui assez simple : nous voulons redéfinir le gestionnaire d'erreur de PHP
Le couplage est moins fort, le sujet n'aura pas besoin dans notre cas de passer une référence de lui même ($this) aux observateurs.
Grace à set_error_handler(), PHP va passer ses erreurs à notre classe. Celle-ci va alors se charger de les enregistrer pour nous.
Voyons un bref exemple :

index.php, autoload supposé activé

```
<?php
$errorHandler = new ErrorHandler;

set_error_handler(array($errorHandler,'error'));

echo $arr[0]; // générons une erreur
```

Notre classe ErrorHandler peut ressembler à ceci :

```
<?php
class ErrorHandler
{
	public function error($errno, $errstr, $errfile, $errline)
	{
		if(error_reporting() == 0)
		{
			return;
		}
		if (!fp = @fopen('my/file','a+'))
		{
			throw new Exception('Impossible d\'ouvrir le fichier de log');
		}
		$message = $errstr . ', ' . $errfile . ', ' . $errline;
		@fputs($fp,$message . PHP_EOL);
		return false; // PHP 5.2 : false doit être retourné pour peupler $php_errormsg
	}
}
```

Ce code simple enregistre l'erreur rencontrée dans un fichier. Notez qu'il doit s'agir d'une erreur PHP, les exceptions, elles, sont traitées à part, par un autre gestionnaire.
Si une erreur PHP intervient dans le gestionnaire d'erreur, PHP est capable de s'en sortir tout seul, néanmoins, nous passons ces erreurs sous silence.
Devant fopen(), et fputs(), nous avons mis un arobase. Celui-ci ne fait qu'une seule chose : pour la commande où il est utilisé, il met (temporairement donc) le rapport d'erreur à off.

L'erreur est donc passée sous silence, mais notre gestionnaire, lui, va quand même l'intercepter. Nous vérifions donc si le rapport d'erreur est à 0. Dans notre cas, cela signifiera qu'une erreur interne à notre classe a été détectée, elle ne sera donc pas traitée par celle-ci.
Evidemment, lorsqu'on fabrique un système qui gère les erreurs, il vaut mieux qu'il en soit exempt lui-même... (il n'y a pas de boucle infinie, PHP sait gérer ce cas là)
De plus, si nous voulons que la variable $php_errormsg soit remplie (dans le cas où track_errors est à On dans le php.ini), alors la méthode de traitement des erreurs, error(), doit retourner false. C'est la documentation qui dit ceci hein ;-)

### 2.1. Problème : gestion du changement

Bien, que se passe-t-il si je veux, en plus d'enregistrer les erreurs dans un fichier, les mémoriser dans une base de données? les envoyer par mail? les afficher à l'écran? les enregistrer dans un tableau php?
Voyons ceci :

```
class ErrorHandler
{
	public function error($errno, $errstr, $errfile, $errline)
	{
		if(error_reporting() == 0)
		{
			return;
		}
		if (!fp = @fopen('my/file','a+'))
		{
			throw new Exception('Impossible d\'ouvrir le fichier de log');
		}
		$pdo = new PDO("mysql:host=localhost;dbname=mydb",'julien','secret');
		$pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

		$message = $errstr . ', ' . $errfile . ', ' . $errline;

		$pdo->exec("INSERT INTO matable (`macol`) VALUES('{$message}')");
		@fputs($fp,$message . PHP_EOL);
		@mail('julien@mail.com','erreur applicative', $message);
		return false; // PHP 5.2 : false doit être retourné pour peupler $php_errormsg
	}
}
```

Le problème semble clair, la classe ErrorHandler possède trop de responsabilités, le code est peu cohésif : les opérations ne sont pas harmonieuses et partent dans tous les sens. Ainsi, il n'est pas simple de changer un élément, et encore moins de tester ce code !

### 2.2. Solution : L'observateur

Une solution possible est d'utiliser un design pattern observateur.
La classe ErrorHandler gère des évènements (ici un seul : la capture d'une erreur PHP), ce n'est pas à elle de les enregistrer, où que ce soit.
Notre classe va agir comme sujet, elle est écoutable, observable.
Des observateurs vont venir se greffer dessus et s'enregistrer sur l'évènement, l'application sera plus découplée, plus cohésive (les responsabilités mieux mises en évidence), mieux testable, et elle gèrera mieux le changement.

Pour ceci nous allons faire appel à 2 interfaces : observeur, et observable. Il aurait été possible, voire judicieux d'utiliser des classes abstraites et de les hériter, mais en PHP, on ne peut hériter que d'une seule classe. Ainsi si notre classe ErrorHandler devait déja hériter (ce qui est dans la pratique assez courant), elle ne pourrait plus :

notre nouveau fichier index

```
<?php
$errorHandler = new ErrorHandler;

$errorHandler->add(new FileWriter(dirname(__FILE__).'/log.txt'));

set_error_handler(array($errorHandler,'error'));

echo $arr[0]; // générons une erreur PHP volontaire (notice)
```

ErrorHandler peut donc recevoir des observateurs

observable

```
<?php
Interface Observable
{
	function add(Observer $obs);
	function del(Observer $obs);
	function notifyObservers();
}
```

observeur

```
<?php
Interface Observer
{
	function update($message);
}
```

notre nouvelle classe ErrorHandler, observable

```
<?php

class ErrorHandler implements Observable
{
	private $_errno;
	private $_errstr;
	private $_errline;
	private $_errfile;
	private $_observers = array();

	public function error($errno, $errstr, $errfile, $errline)
	{
		if(error_reporting() == 0)
		{
			return;
		}
		$this->_errno   = $errno;
		$this->_errstr  = $errstr;
		$this->_errfile = $errfile;
		$this->_errline = $errline;
		$this->notifyObservers();
		return false; // PHP 5.2 : false doit être retourné pour peupler $php_errormsg
	}

	private function getError()
	{
		return $this->_errstr . ', ' . $this->_errfile . ', ' . $this->_errline;
	}
```

```
	public function add(Observer $obs)
	{
		$this->_observers[] = $obs;
		return $this;
	}

	public function del(Observer $obs)
	{
		if (is_int($key =
array_search($obs,$this->_observers)))
		{
			unset($this->_observers[$key]);
		}
		return $this;
	}

	public function notifyObservers()
	{
		foreach ($this->_observers AS $observer)
		{
			try{
				$observer->update($this-
>getError()); // délégation
			}catch(Exception $e){
				die($e->getMessage());
			}
		}
	}
}
```

ErrorHandler doit pouvoir s'ajouter des observateurs (qui vont la "guetter"), s'en supprimer, et les notifier.
A chaque erreur, on utilise **notifyObservers()**, qui va boucler sur tous les observateurs, et leur passer le message d'erreur.
A eux d'en faire ce qu'ils veulent, ça n'est plus du ressort de ErrorHandler.
Le couplage reste le même, mais la cohésion est fortement améliorée. A chaque objet, son rôle, et sa responsabilité.
Le code de la classe ErrorHandler est bien plus facilement testable. De plus, on peut ajouter n'importe quoi comme observateur.
Le fait que **add()** retourne $this va nous permettre de chainer les méthodes, regardez plutot :

observateur fichiers

```
<?php
class FileWriter implements Observer
{
	private $_fp;

	public function __construct($filepath)
	{
		if (FALSE === $this->_fp =
@fopen($filepath,'a+'))
		{
			throw new Exception('Impossible
d\'ouvrir le fichier de log');
		}
	}

	public function update($message)
	{
		@fputs($this->_fp,$message . PHP_EOL);
	}
}
```

observateur base de données

```
<?php
class BDDWriter implements Observer
{
	private $_pdo;
	private $_table;
	private $_col;

	public function
__construct($host,$login,$pass,$dbname,$table,$col)
	{
		$this->_pdo = new
PDO("mysql:host=$host;dbname=$dbname",$login,$pass);
		$this->_pdo-
>setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION)
;
		$this->_col   = (string)$col;
		$this->_table = (string)$table;
	}

	public function update($message)
	{
		$this->_pdo->exec("INSERT INTO $this-
>_table (`$this->_col`) VALUES('{$message}')");
	}
}
```

Chaque observateur est testable de manière unique, il n'a besoin de personne.
Chaque observateur implémente sa propre configuration, une base de données necessite des indentifiants (entres autres), un fichier necessite une chemin pour le trouver.
Nous injectons après simplement chaque observateur à notre observable ErrorHandler :

injection d'observateurs configurés

```
$errorHandler = new ErrorHandler;

$errorHandler->add(new
FileWriter(dirname(__FILE__).'/log.txt'))
		->add(new
BDDWriter('localhost','root','','test','test','nom'));

set_error_handler(array($errorHandler,'error'));

echo $arr[0]; // générons une erreur pour tester
```

## 2.3. Allons plus loin : d'autres observateurs

Effet immédiat : je peux réutiliser l'observateur fichier (FileWriter) pour lui demander d'écrire vers la sortie standard (php://output), ceci grace aux contextes de flux de PHP5 (une couche d'abstraction supplémentaire très intéréssante).
Autre effet de la séparation des rôles : je peux encore ajouter un observateur, email par exemple
Et même, pour les tests, je peux rajouter un observateur Mock : c'est un observateur qui va simplement stocker dans un tableau PHP, les erreurs déléguées par ErrorHandler. Il me sera possible de les afficher après, regardez plutot :

un observateur email simple

```
<?php
class MailWriter implements Observer
{
	private $_to;
	const SUBJECT = 'erreur signalée';

	public function __construct($to)
	{
		$this->_to = (string)$to;
```

```
                if(filter_var($this->_to,FILTER_VALIDATE_EMAIL) === false)
                {
                        throw new Exception('Adresse email non conforme');
                }
        }

        public function update($message)
        {
                @mail($this->_to,self::SUBJECT, $message);
        }
}
```

observateur mock

```
<?php
class mockWriter implements Observer
{
        private $_messages = array();

        public function update($message)
        {
                $this->_messages[] = $message;
        }

        public function show()
        {
                print_r(new ArrayObject($this->_messages));
        }
}
```

L'implémentation complète est très intuitive :

```
<?php
$errorHandler = new ErrorHandler;

$errorHandler->add(new FileWriter(dirname(__FILE__).'/log.txt'))
             ->add(new BDDWriter('localhost','root','','test','test','nom'))
             ->add(new MailWriter('julien@emailme.com'))
             ->add($mock = new mockWriter());

set_error_handler(array($errorHandler,'error'));

echo $arr[0]; // générons une erreur

$mock->show(); // et affichons là ;)
```

## 2.4 Plus de potenciel : la SPL entre en jeu

Il est immédiatement remarquable que notre sujet observable (ErrorHandler), aggrège des objets observateurs divers.
Pourquoi ne pas utiliser la SPL, et l'interface Iterator pour pouvoir facilement lister les observateurs attachés ?
Et même mieux encore, utilisons IteratorAggregate, nous n'aurons pas besoin de définir toutes les méthodes de l'itérateur dans la classe ErrorHandler comme ceci.
Au lieu de cela, il faudra définir une méthode getIterator(), qui va retourner l'objet d'itération. Et encore vraiment mieux : utilisons un ArrayObject, qui contient déja tout ce qu'il faut !

ErrorHandler complet, itérable

```
<?php
class ErrorHandler implements Observable, IteratorAggregate
{
        private $_errno;
        private $_errstr;
        private $_errline;
        private $_errfile;
        private $_observers;

        public function __construct()
        {
                $this->_observers = new ArrayObject();
        }

        public function error($errno, $errstr, $errfile, $errline)
        {
                if(error_reporting() == 0)
                {
                        return;
                }
                $this->_errno   = $errno;
                $this->_errstr  = $errstr;
                $this->_errfile = $errfile;
                $this->_errline = $errline;
                $this->notifyObservers();
                return false; // PHP 5.2 : false doit être retourné pour peupler $php_errormsg
        }

        private function getError()
        {
                return $this->_errstr . ', ' . $this->_errfile . ', ' . $this->_errline;
        }

        public function add(Observer $obs)
        {
                $this->_observers[] = $obs;
                return $this;
        }

        public function del(Observer $obs)
        {
                if (is_int($key = array_search($obs,$this->_observers)))
                {
                        unset($this->_observers[$key]);
                }
                return $this;
        }

        protected function notifyObservers()
        {
                // $this est intercepté par l'itérateur
                foreach ($this AS $observer)
                {
                        try{
                                $observer->update($this->getError());
                        }catch(Exception $e){
                                die($e->getMessage());
                        }
                }
        }

        public function getIterator()
        {
                return $this->_observers;
        }
}
```

Remarquez comment ErrorHandler profite de l'itérateur lorsqu'il

s'agit d'itérer sur tous les observateurs.
Ceci peut aussi se faire à l'exterieur de la classe :

index qui itère sur les observateurs de ErrorHandler

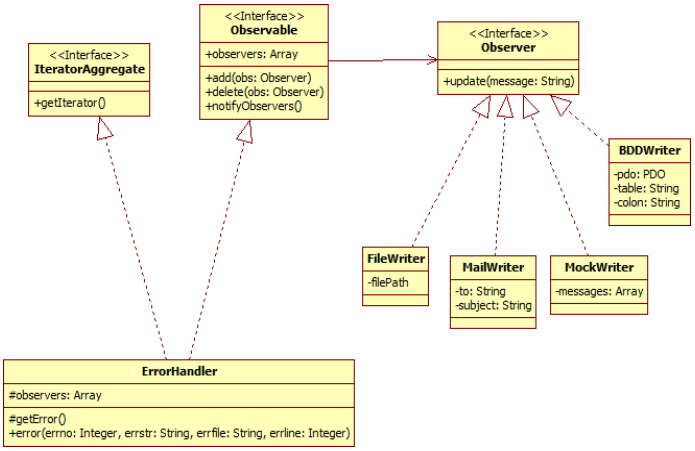
```
<?php
$errorHandler = new ErrorHandler;

$errorHandler->add(new
FileWriter(dirname(__FILE__).'/log.txt'));
$errorHandler->add(new
BDDWriter('localhost','root','','test','test','nom'));
$errorHandler->add(new FileWriter('php://output'));
$errorHandler->add(new
MailWriter('julien@anaska.com'));
$errorHandler->add($mock = new mockWriter());

set_error_handler(array($errorHandler,'error'));

foreach ($errorHandler as $writer) {
       echo get_class($writer); // facile tout de même
non?
}
```

Très simple, testable, découplé, efficace, réutilisable; J'espère que ca vous donne des idées ;-)




## 3. Conclusion

Nous venons de voir comment utiliser un design pattern observeur - observable. Celui-ci est utlisable dès que le couplage (les dépendances) devient trop fort.
Il participe à la séparation des rôles et au découplage applicatif. Ici, le design pattern est relativement lâche, car les objets n'ont qu'un couplage faible.
ErrorHandler necessite des observers, mais pas le contraire. En général, le couplage est très sérré, et il y a un lien de dépendance dans les 2 sens, si bien que l'on définit alors l'oberveur avec une méthode update(Observable $obs).
C'est le cas de l'interface SplObserver (car la SPL propose déja un design pattern observateur, regardez ici)

De la même manière, utiliser des classes abstraites aurait permis de passer notifyObservers() en private, ce qui est préférable.

Retrouvez le tutoriel de Julien Pauli en ligne : Lien2

# Les cadres en CSS

Cet article est la traduction de **CSS Frames v2, full-height** qui décrit une technique pour réaliser des cadres (frames) avec du CSS.

## 1. Introduction

Il y a déjà quelques temps, en août 2003, j'ai écrit un court article intitulé CSS Frames (Lien3), dans lequel j'ai décrit une technique permettant de simuler, en apparence, un cadre HTML avec du CSS. C'était il y a plus de trois ans, et je pense donc qu'il est temps de revoir et améliorer quelque peu la technique, d'autant plus que la page de démonstration d'origine pour les cadres CSS (Lien4) a reçu, récemment, de nombreuses visites.

J'ai seulement jeté un coup d'oeil rapide au code, et je peux affirmer qu'il date de quelques années. C'est parfois embarassant de se replonger du vieux code. La démo contient une liste des navigateurs avec lesquels j'avais fait mes essais, "Mozilla Firebird 0.61" y apparaît. Comme je le disais, il commence à dater, donc voilà la mise à jour.

La raison qui justifie la création d'une technique de cadres CSS est, bien sûr, la volonté d'abandonner l'utilisation des cadres HTML. Si vous n'en connaissez pas les raisons, lisez mon article Who framed the web: Frames and usability (Lien5).

Maintenant, plongeons-nous dans la mise à jour de la technique des cadres CSS.

## 2. La mise en page : entête, corps de page pleine hauteur, pied de page

La disposition que je décris ici aura une entête fixe en haut de la page, une zone (au milieu) avec un contenu pouvant défiler, et un pied de page fixe en bas de page. Le concept peut être adapté à d'autres types de disposition en cadres, mais je vous le laisse en exercice.

Je me suis aussi assuré que le corps de la page prenne l'intégralité de la hauteur visible même s'il n'y a pas assez de contenu pour activer le défilement. Pour cela, j'ai utilisé la technique décrite par Peter-Paul Koch dans 100% height (Lien6).

La mise en page repose sur trois blocs : #header, #content et #footer. L'effet de cadre se crée en deux étapes. Tout d'abord, on donne des positions fixes à #header et #footer, respectivement en haut et en bas de la page visible. Ensuite, on fixe des marges intérieures (padding) en haut et en bas de #content, afin qu'elles correspondent aux hauteurs de #header et #footer.

L'ensemble des trois blocs sont réunis dans un élément div ayant pour id #wrap, afin de faciliter le contrôle de la largeur d'affichage et du centrage horizontal. #content est inclus dans #content-wrap à cause de la technique des 100% en hauteur que j'utilise. Le bloc qui défile a besoin d'avoir une hauteur de 100%, mais a également un padding haut et bas. Cela devrait s'ajouter et faire plus de 100%, c'est pourquoi le padding est défini au niveau de #content à la place, étant donné que les deux ont le même arrière-plan.

## 3. Différentes largeurs

Selon que vous souhaitiez une mise en page qui s'ajuste à la largeur visible ou non, vous aurez besoin d'utiliser différentes largeurs pour les éléments #header et #footer. Les éléments qui ont la propriété position:fixed sont positionnés en respectant la fenêtre (Lien7), donc le fait de mettre leur largeur à 100% (vous aurez besoin de fixer une largeur ou de les réduire pour coller à leur contenu) et ajuster la largeur de #wrap ne fonctionnera pas. Les blocs #header et #footer seraient toujours ajuster à la largeur de la fenêtre. À moins que l'on spécifie un left ou un right, leur coté gauche sera à gauche de #wrap, où ils seraient s'ils l'ont ne les avaient pas positionnés du tout.

J'ai préparé deux exemples pour montrer la différence. Dans le premier exemple (Lien8), l'agencement est très fluide, et donc #header et #footer occupent toute la largeur. Dans le second exemple (Lien9) la largeur de la page est fixée à 40 em, et les largeurs de #header et #footer ont donc la même valeur.

## 4. La feuille de style CSS

Voici le CSS que j'ai utilisée dans le premier exemple :

Code CSS

```
html, body {
  margin:0;
  padding:0;
  height:100%; /* 100 % en hauteur */
}

html>body #wrap {height:100%;} /* 100 % en hauteur */

#header {
   width:100%;
   height:5em;
}

html>body #header {
  position:fixed;
  z-index:10; /* empêche certains problèmes avec les
élements de formulaire */
}

html>body #content-wrap {height:100%;} /* 100 % en
hauteur */

html>body #content {padding:6em 1em;} /* 6em = hauteur
de #header et #footer + 1em, 1em = donne au contenu un
peu d'espace par rapport aux bords */

#footer {
   width:100%;
   height:5em;
}

html>body #footer {
```

```
  position:fixed;
  bottom:0;
  z-index:10; /* empêche certains problèmes avec les
élements de formulaire */
}
```

Et maintenant, voici comment fonctionne le second exemple :

Code CSS

```
html, body {
   margin:0;
   padding:0;
   height:100%; /* 100 % en hauteur */
}

html>body #wrap {height:100%;} /* 100 % en hauteur */

#wrap {
   width:40em;
   margin:0 auto;
}

#header {
   width:40em;
   height:5em;
}

html>body #header {
   position:fixed;
   z-index:10; /* empêche certains problèmes avec les
élements de formulaire */
}

html>body #content-wrap {height:100%;} /* 100 % height
*/

html>body #content {padding:6em 1em;} /* 6em = hauteur
de #header et #footer + 1em, 1em = donne au contenu un
peu d'espace par rapport aux bords */

#footer {
   width:40em;
   height:5em;
}

html>body #footer {
   position:fixed;
   bottom:0;
   z-index:10; /* empêche certains problèmes avec les
élements de formulaire */
}
```

## 5. Les ajustements pour Internet Explorer (IE)

Malheureusement, Internet Explorer sous Windows dans des versions inférieures ou égales à 6 ne supportent pas la propriété CSS position:fixed. Comme il est utilisé par trop de monde, on ne peut l'ignorer, et il est nécessaire de placer quelques rustines. Etonnement, elles sont inutiles sous IE7, ça fonctionne tel quel ! Toutefois, pour IE5 et 6, il faudra ajouter quelques règles supplémentaires à l'aide d'un commentaire conditionnel :

Code (X)HTML

```
<!--[if lt IE 7]>
<link rel="stylesheet" href="ie.css" type="text/css">
<![endif]-->
```

Le fichier ie.css contient le code CSS qui suit :

Code CSS

```
html, body {background:url(foo) fixed;}

#header, #footer {
   position:absolute;
   z-index:10;
}
#header {
    top:expression(eval(document.compatMode &&
document.compatMode=='CSS1Compat') ?
documentElement.scrollTop : document.body.scrollTop)
}

#wrap, #content-wrap {height:100%;}

#content {padding:6em 1em;}

#footer {
    top:expression(eval(document.compatMode &&
document.compatMode=='CSS1Compat') ?
    documentElement.scrollTop
+(documentElement.clientHeight-this.clientHeight) :
    document.body.scrollTop
+(document.body.clientHeight-this.clientHeight));
}
```

Ceci fonctionne avec IE 5, 5.5 et 6, bien que je ne me sois pas embêté à recentrer horizontalement sous IE 5.* ; en effet, il est temps d'abandonner les IE 5.*

Le défilement est quelque peu instable, mais il fonctionne. Étant donné que les expressions font appel à du JavaScript, cela ne fontionnera pas s'il est désactivé. Heureusement, dans de tels cas, cela entrainera juste le défilement de toute la page, et aucun contenu sera caché.

## 6. Améliorations progressives et jolies dégradations

Dans les navigateurs ne supportant pas position:fixed (à l'exception de IE5-6), la page entière défilera, ce qui est un défaut tout à fait acceptable. J'utilise un "sélecteur enfant" afin de permettre uniquement aux navigateurs le supportant de voir les déclarations position:fixed. Vouloir laisser les navigateurs ne supportant pas le positionnement fixe (à l'exception de IE5-6 sous Windows) voir l'ensemble des propriétés CSS constitue un autre problème.

La technique fonctionne sur tous les navigateurs modernes qui sont passés entre mes mains, à une exception. L'astuce utilisée pour ajuster la zone de contenu sur l'intégralité de la hauteur de la fenêtre provoque un comportement étrange avec les navigateurs ICab et Internet Explorer pour Mac. Ce sont des problèmes esthétiques d'une gravité assez faible, je pense donc qu'il est possible d'y survivre.

À part cela, je ne suis pas conscient d'autres problèmes. Cependant cela ne signifie pas qu'il n'y en ait pas, donc je vous remercie de m'avertir si vous trouvez des bugs, ou si vous souhaitez proposer quelques améliorations supplémentaires.

Retrouvez l'article de Roger Johansson traduit par Nicolas Vallée en ligne : Lien10

# Les listes de définitions : mal utilisées ou mal comprises ?

Cet article est la traduction de **Definition lists - misused or misunderstood?** qui a pour but de vous montrer l'utilité et le type d'utilisation des listes de définitions.

## 1. Introduction

Qu'est ce que les listes de définitions ? Quand sont elles appropriées ? Et comment les personnaliser pour qu'elles ressemblent à des tableaux, des galeries d'images, des calendriers, etc.

## 2. Qu'est-ce que les listes de définitions ?

Les listes de définitions sont constituées de deux parties : un terme et une description. Pour écrire une liste de définitions, vous avez besoin de trois éléments HTML :

- Un conteneur <dl> ;
- Un terme de définition <dt> ;
- Une description de définition <dd>.

Par exemple :

Simple liste de définitions

```
<dl>
  <dt>Grenouille</dt>
  <dd>Chose verte humide</dd>
  <dt>Lapin</dt>
  <dd>Chose chaude et douce</dd>
</dl>
```

Vous pouvez utiliser une multitude de <dt> et <dd> dans une liste de définitions.

Liste avec plusieurs dd

```
<dl>
  <dt>Punt</dt>
  <dd>Shooter un ballon</dd>
  <dd>Prendre un paris</dd>
</dl>
```

Liste avec plusieurs dt

```
<dl>
  <dt>HTML</dt>
  <dt>XHTML</dt>
  <dd>Tout sauf le CSS</dd>
</dl>
```

Vous pouvez aussi utiliser des éléments de type bloc dans les descriptions de définitions, comme les balises <p> et <ul>. Par contre, vous ne pouvez pas utiliser d'éléments de type bloc dans les termes de définition.

Liste avec une balise p

```
<dl>
  <dt>Grenouille</dt>
  <dd><p>Chose verte humide et qui croasse.</p></dd>
</dl>
```

Liste avec une sous-liste ul

```
<dl>
  <dt>Grenouille</dt>
  <dd>
    <ul>
      <li>Humide</li>
      <li>Vert</li>
      <li>Croasser</li>
    </ul>
  </dd>
</dl>
```

## 3. Quand les listes sont-elles appropriées ?

Il y a deux points de vue à propos de leur utilisation.

Certains pensent qu'elles ne devraient être utilisées que pour des mots et leurs définitions. Et d'autres pensent qu'elles devraient être utilisées pour mettre en évidence les élements directement liés entre eux (comme les couples nom/valeur).

Le deuxième point de vue est appuyé d'un exemple qui respecte les normes du W3C.

Une autre application des listes de définitions, par exemple, est la mise en évidence de dialogues avec les balises <dt> désignant l'orateur, et les balises <dd>, ses propos.

Les listes dans les documents HTML (Lien11)

Bien que certains ne soient pas d'accord avec cet exemple, il suggère que les listes de définitions peuvent être utilisées au delà du simple fait terme - définition, du moment qu'il y a une relation entre chaque éléments. Suivant cet argument, tous les exemples ci-dessous peuvent être mis en évidence à l'aide de listes de définitions :

DT: Orateur
    DD: Citation

DT: Image
    DD: Description
    DD: Localisation
    DD: Photographe

DT: Terme
    DD: Image descriptive
    DD: Description

DT: Site web (lien)
    DD: Description

DT: Date
    DD: Évênement

DT: Évênement
    DD: Date

DD: Description
DD: Lieu

DT: Lien interne
DD: Accueil
DD: Section 1
DD: Section 2

DT: Liens externes
DD: Lien externe 1
DD: Lien externe 2

## 4. Y'a-t-il des inconvénients à utiliser les listes de définitions ?

Avant de les utiliser, vous devez être conscient que ce n'est pas la meilleure option dans tous les cas.

Le <dt> ne peut pas contenir d'éléments de type bloc, et plus particulièrement les <hN> (exemple : h1). Si le contenu d'un <dt> ne peut pas être marqué comme un titre, dans ce cas, il ne peut pas avoir la même importance dans la hiérarchie du document. De même, Google et d'autres moteurs de recherche ne référenceront pas les listes de définitions comme d'autres pages basées sur le système de titrage.

Bien que les listes de définitions puissent être formatées pour ressembler aux données tabulaires, elles ne peuvent pas contenir des caractéristiques d'accessibilités comme les étiquettes (label) et les titres (h1, etc.) pour lier les informations entre elles. C'est pourquoi, elles ne devraient pas être utilisées pour remplacer des données tabulaires complexes.
IV. Y'a-t-il des inconvénients à utiliser les listes de définitions ?
Avant de les utiliser, vous devez être conscient que ce n'est pas la meilleure option dans tous les cas.

Le <dt> ne peut pas contenir d'éléments de type bloc, et plus particulièrement les <hN> (exemple : h1). Si le contenu d'un <dt> ne peut pas être marqué comme un titre, dans ce cas, il ne peut pas avoir la même importance dans la hiérarchie du document. De même, Google et d'autres moteurs de recherche ne référenceront pas les listes de définitions comme d'autres pages basées sur le système de titrage.

Bien que les listes de définitions puissent être formatées pour ressembler aux données tabulaires, elles ne peuvent pas contenir des caractéristiques d'accessibilités comme les étiquettes (label) et les titres (h1, etc.) pour lier les informations entre elles. C'est pourquoi, elles ne devraient pas être utilisées pour remplacer des données tabulaires complexes.

## 5. Personnalisation de listes de définitions

Voici des exemples de personnalisation :

### 5.1. Liste de définitions sans style

Code (X)HTML
```
<dl>
  <dt>Grenouille verte et jaune</dt>
  <dd>Nisl ut aliquip ex ea commodo consequat</dd>
  <dt>Crapaud buffle</dt>
  <dd>Lorem ipsum dolor sit amet elit...</dd>
  <dt>Grenouille tachetée</dt>
  <dd>Facilisis at vero eros et accumsan</dd>
</dl>
```

Liste de définitions sans styles : Lien12

### 5.2. Liste de définitions avec un style basique

Code (X)HTML
```
<dl class="margins-removed">
  <dt>Grenouille verte et jaune</dt>
  <dd>Nisl ut aliquip ex ea commodo consequat</dd>
  <dt>Crapaud buffle</dt>
  <dd>Lorem ipsum dolor sit amet elit...</dd>
  <dt>Grenouille tachetée</dt>
  <dd>Facilisis at vero eros et accumsan</dd>
</dl>
```

Code CSS
```
dl.margins-removed{
  margin: 0;
  padding: 0;
}

.margins-removed dt{
  margin: 0;
  padding: 0;
  font-weight: bold;
}

.margins-removed dd{
  margin: 0 0 1em 0;
  padding: 0;
}
```

Liste de définitions sans marge : Lien13

### 5.3. Liste de définitions avec image de fond

Code (X)HTML
```
<dl class="background-image">
  <dt>Grenouille verte et jaune</dt>
  <dd>Nisl ut aliquip ex ea commodo consequat</dd>
  <dt>Crapaud buffle</dt>
  <dd>Lorem ipsum dolor sit amet elit...</dd>
  <dt>Grenouille tachetée</dt>
  <dd>Facilisis at vero eros et accumsan</dd>
</dl>
```

Code CSS
```
.background-image dt{
  color: #000;
  font-weight: bold;
  padding: 0;
}

.background-image dd{
  margin: 0 0 1em 0;
  padding: 0 0 0 10px;
  background-image: url('arrow.gif');
  background-repeat: no-repeat;
  background-position: 0 .5em;
}
```

<dd> avec une image de fond : Lien14

### 5.4. Liste de définitions sous forme de boîte

Code (X)HTML

```
<dl class="border-around">
  <dt>Grenouille verte et jaune</dt>
  <dd>Nisl ut aliquip ex ea commodo consequat</dd>
  <dt>Crapaud buffle</dt>
  <dd>Lorem ipsum dolor sit amet elit...</dd>
  <dt>Grenouille tachetée</dt>
  <dd>Facilisis at vero eros et accumsan</dd>
</dl>
```

Code CSS

```
dl.border-around{
  margin: 2em 0;
  padding: 0;
  width: 20em;
}

.border-around dt{
  background-color: #131210;
  color: #959289;
  padding: .5em .5em;
  font-weight: bold;
  text-align: center;
  text-transform: uppercase;
  border-left: 1px solid #131210;
  border-right: 1px solid #131210;
  border-top: 1px solid #131210;
}

.border-around dd
{
  margin: 0 0 1em 0;
  background: #DBD8D8;
  text-align: center;
  padding: 1em .5em;
  font-style: italic;
  border-left: 1px solid #131210;
  border-right: 1px solid #131210;
  border-bottom: 1px solid #131210;
}
```

Liste de définitions sous forme de boite : Lien15

### 5.5. Formater une liste comme un tableau

Code (X)HTML

```
<dl class="table-display">
  <dt>Grenouille verte et jaune</dt>
  <dd>Nisl ut aliquip ex ea commodo consequat</dd>
  <dt>Crapaud buffle</dt>
  <dd>Lorem ipsum dolor sit amet elit...</dd>
  <dt>Grenouille tachetée</dt>
  <dd>Facilisis at vero eros et accumsan</dd>
</dl>
```

Code CSS

```
dl.table-display{
  width: 41.1em;
  margin: 2em 0;
  padding: 0;
  font-family: georgia, times, serif;
}

.table-display dt{
  width: 15em;
  float: left;
  margin: 0 0 0 0;
  padding: .5em;
  border-top: 1px solid #999;
  font-weight: bold;
}

/* commented backslash hack for mac-ie5 \*/
dt { clear: both; }
/* end hack */

.table-display dd{
  float: left;
  width: 24em;
  margin: 0 0 0 0;
  padding: .5em;
  border-top: 1px solid #999;
}
```

Représentation tabulaire : Lien16

### 5.6. Liste flottante avec une image et sa description

Code (X)HTML

```
<dl class="float-right">
  <dt>Fleur de Banksia</dt>
  <dd><img src="flower.jpg" alt=""></dd>
  <dd><em>Banksius maximus</em></dd>
  <dd>On la trouve au large des côtes</dd>
</dl>
```

Code CSS

```
dl.float-right{
  border: 1px solid #000;
  background-color: #ddd;
  width: 142px;
  text-align: center;
  padding: 0 0 10px 0;
  float: right;
  margin: 0 0 1em 1em;
}

.float-right dt{
  font-weight: bold;
  background-color: #131210;
  color: #959289;
  padding: 5px 10px;
  margin-bottom: 10px;
}

.float-right dd img{
  border: 1px solid #000;
  width: 100px;
  height: 100px;
}

.float-right dd{
  margin: 0;
  padding: 0 10px 5px 10px;
  font-size: 85%;
}
```

Liste de définitions floatante avec une image et sa description : Lien17

### 5.7. Formater une liste comme une galerie d'images

Code (X)HTML

```
<dl class="gallery">
  <dt><img src="flower.jpg" alt=""></dt>
  <dt>Ici le titre</dt>
  <dd>Ici la description</dd>
</dl>
<dl class="gallery">
  <dt><img src="flower2.jpg" alt=""></dt>
  <dt>Ici le titre</dt>
  <dd>Ici la description</dd>
</dl>
<dl class="gallery">
  <dt><img src="flower3.jpg" alt=""></dt>
  <dt>Ici le titre</dt>
  <dd>Ici la description</dd>
</dl>
```

Code CSS

```
dl.gallery{
  border: 1px solid #000;
  background-color: #ddd;
  width: 102px;
  text-align: center;
  padding: 10px;
  float: left;
  margin-right: 1em;
}

.gallery dt { font-weight: bold; }

.gallery dt img{
  border: 1px solid #000;
  width: 100px;
  height: 100px;
}

.gallery dd{
  margin: 0;
  padding: 0;
}
```

Liste de définitions sous forme de galerie d'images : Lien18

### 5.8. Formater comme une liste d'évènements

Code (X)HTML

```
<dl class="event">
  <dt>23 March</dt>
  <dd>Rencontre du club automobile</dd>
  <dd>19h00</dd>
  <dd>
    Voir une large gamme de voitures classiques, le
tout en un seul lieu. Il y aura aussi des ballons, des
clowns, châteaux gonflables et de la nourriture.
    Du plaisir pour toute la famille.
  </dd>
  <dt>13 June</dt>
  <dd>Foire aux gâteau</dd>
  <dd>12h00</dd>
  <dd>Gâteaux de toutes sortes. Tous les fonds
recueillis sont versés à la fondation "Minky
Whale".</dd>
</dl>
```

Code CSS

```
dl.event{
  margin: 2em 0;
  padding: 0;
  font-family: georgia, times, serif;
}

.event dt{
  position: relative;
  left: 0;
  top: 1.1em;
  width: 5em;
  font-weight: bold;
}

.event dd{
  border-left: 1px solid #000;
  margin: 0 0 0 6em;
  padding: 0 0 .5em .5em;
}
```

Liste sous forme de calendrier d'évènements : Lien19

### 5.9. Formater comme un tableau à multiples DD

Code (X)HTML

```
<dl>
  <dt>Grenouille verte et jaune</dt>
  <dd class="first">Nisl ut aliquip ex ea commodo
consequat</dd>
  <dd>Lorem ipsum dolor sit amet elit, sed diam nonummy
nibh euismod tincidunt</dd>
  <dt>Crapaud buffle</dt>
  <dd class="first">Lorem ipsum dolor sit amet elit,
sed diam nonummy nibh euismod tincidunt</dd>
  <dd>Nisl ut aliquip ex ea commodo consequat</dd>
</dl>
```

Code CSS

```
dl { border-bottom: 1px solid #999; }

dt{
  width: 15em;
  padding: .5em;
  float: left;
  margin: 0;
  border-top: 1px solid #999;
  font-weight: bold;
}

dd{
  margin-left: 16em;
  padding: .5em;
}

dd.first { border-top: 1px solid #999; }
```

Liste sous forme d'un tableau à plusieurs <dd> : Lien20

Retrouvez l'article de Russ Weakley traduit par Joris Crozier en ligne : Lien22

# Développer un calendrier dynamique avec AJAX et PHP

L'avènement du "web 2.0" avec notamment ses interfaces riches a placé l'AJAX au centre des applications web. Pour faciliter le développement, de nombreux framework ont vu le jour comme les projets prototype et openrico. Au travers d'une application "calendrier", nous allons tenter ici de les mettre en oeuvre.

## 1. Présentation du projet "calendrier"

Tout au long de cet article, nous allons développer un calendrier qui s'affichera mois par mois à la manière d'un "google calendar" ou de ce que l'on peut faire avec l'api Yahoo. Nous verrons alors comment peupler ce calendrier avec les dates à l'aide d'AJAX et de la librarie prototype.

Ensuite, pour chaque date, nous pourrons afficher des évènements stockés dans une base de données.

Enfin, nous mettrons en forme notre calendrier à l'aide du framework javascript openrico.

Les langages utilisés seront donc javascript et PHP.

Et aussi voir un exemple du script sur cette page de démonstration (Lien).

Utiliser un framework est un choix que je vous encourage à faire si vous le pouvez ! Vous bénéficierez généralement de plus de souplesse, de qualité et éviterez de nombreux casse-tête quant à la compatibilité du code avec les navigateurs.

### 1.1. Pré-requis

Vous trouverez dans les paragraphes suivants la liste des outils nécessaires à la création de cette application.

#### 1.1.1. PHP MYSQL

Pour notre projet, nous avons besoin de PHP pour les traitements et d'une base de données MYSQL pour stocker les évènements liés à chaque date.
Pour se faire nous allons donc déployer WAMP sur une station windows.

Télécharger WAMP : Lien22

Une fois les serveurs en route, vous pouvez créer la base de données qui contiendra les évènements associés à chaque jour. Voici un jeu d'essai que vous pouvez utiliser pour tester l'application :

```
CREATE DATABASE `calendrier` ;
CREATE TABLE `evenements` (  `evenement_id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,`evenement_date` DATE NOT NULL ,`evenement_comment` TEXT NOT NULL ) TYPE = innodb;
INSERT INTO `evenements` ( `evenement_id` , `evenement_date` , `evenement_comment` )
VALUES (
       '1', '2007-11-04', 'Réunion projet "calendrier"'
       ), (
       '2', '2007-11-04', 'Départ à la retraite de thierry'
);
INSERT INTO `evenements` ( `evenement_id` , `evenement_date` , `evenement_comment` )
VALUES (
       '3', '2007-11-08', 'Démarrage du projet "SEO"'
       ), (
       '4', '2007-12-24', 'Préparation des cadeaux :)'
);
```

#### 1.1.2. Le framework prototype

Le framework prototype propose un ensemble de fonctions javascript facilitant le développement AJAX.
Pour appréhender plus en détails les particularités et le fonctionnement de ce framework, vous pouvez consulter l'article sur "prototype" par Aurélien MILLET
Ce framework est directement intégré dans la bibliothèque openrico, vous n'avez pas donc pas besoin de le télécharger. Une documentation de ce framework est disponible en français :

Documentation prototype (v.1.4.0) : Lien23

#### 1.1.3. La librairie openrico

Openrico est une bibliothèque de fonctions javascript permettant de créer de nombreux effets graphiques ( comme des menus en accordéon, des blocs arrondis ou encore la gestion du drag and drop ).
Ce projet intègre le framework prototype pour la gestion des requêtes AJAX. Télécharger la bibliothèque et décompresser l'archive par exemple dans le répertoire "js" de votre application web

Télécharger OPENRICO (v 2.0, PROTOTYPE inclus) : Lien24

### 1.2. Fonctionnement de l'application

Note application se décomposera en trois fichiers principaux :

- **calendrier.php** : ce fichier va contenir notre calendrier, c'est à dire un tableau de 7 colonnes (pour chaque jour de la semaine) et de 6 lignes.
- **ajax/ajax_calendrier.php** : il aura la charge de peupler le calendrier avec les dates pour un mois de l'année donné en paramètre.
- **ajax/ajax_commentaires.php** : ce fichier aura pour but d'afficher la liste des évènements pour le jour du calendrier que l'on aura choisi.

Dans la pratique, nous allons donc afficher notre calendrier avec la possibilité de naviguer de mois en mois. Pour chaque mois choisi, une requête ajax va être transmise vers le fichier ajax_calendrier.php qui va retourner un tableau contenant tous les jours à afficher dans le calendrier pour le mois demandé.

Cette réponse sous forme de tableau sera en fait au format JSON ! ( qui est un format directement lisible par prototype , on en verra l'intérêt un peu plus tard).

Lorsque l'utilisateur cliquera sur un jour du calendrier, une nouvelle requête ajax va être envoyé via prototype au fichier ajax_commentaires.php. Ce fichier se chargera d'aller chercher dans la base de données les évènements liés au jour choisi et de les renvoyer à prototype pour les afficher à l'écran.

Passons maintenant à la pratique et rentrons dans le code !

Vous noterez que l'on ne fait plus vraiment de l'AJAX , puisque comme son nom l'indique, AJAX implique une réponse au format XML et qu'on va plutôt opter pour un format JSON. Ceci dit, pour des raisons pratiques, je continuerai à parler d'AJAX.

## 2. Développement de notre calendrier

### 2.1. Création du calendrier

calendrier.php (entête)

```
<?php
//connexion à la base de données mysql
$ress_mysql = mysql_connect('servername', 'username',
'');
$db = mysql_select_db('dbname', $ress_mysql) or die
("Connexion impossible");
?>
<!-- On inclut la librairie openrico / prototype -->
<script src="./js/rico/src/prototype.js"
type="text/javascript"></script>
```

On effectue ici deux opérations. La première c'est la connexion à la base de données. Et ensuite, ce qui est déjà plus intéressant, on importe la bibliothèque javascript openrico.
On remarque au passage que la librairie prototype fait partie des fichiers "openrico" à inclure.
Maintenant, on peut se charger d'afficher le tableau html qui va contenir notre calendrier.

calendrier.php (corps calendrier)

```
<div id="calendrier" class="conteneur calendrier"
style="width:260px;background-color:#c6c3de;">
    <table class="tab_calendrier" align="center"
style="margin:auto;background:#ffffff;">
       <tr>
                <td class="titre_calendrier" colspan="7"
width="100%">
                  <a id="link_precedent"
href="#">Precedent</a>
                  <span id="titre"></span>
                  <a id="link_suivant"
href="#">Suivant</a>
                </td>
          </tr>
      <tr>
        <td  class="cell_calendrier" >
        Lun
        </td>
        <td  class="cell_calendrier" >
        Mar.
       </td>
              <td  class="cell_calendrier">
              Mer.
              </td>
              <td  class="cell_calendrier">
              Jeu.
              </td>
              <td  class="cell_calendrier" >
              Ven.
              </td>
              <td  class="cell_calendrier">
              Sam.
              </td>
              <td  class="cell_calendrier">
              Dim.
              </td>
      </tr>
      <?php
             $compteur_lignes=0;
             $total=1;
             while($compteur_lignes<6){
                echo '<tr>';
                $compteur_colonnes=0;
                while($compteur_colonnes<7){
                   echo '<td id="'.$total.'"
class="cell_calendrier" >';
                   echo '</td>';
                   $compteur_colonnes++;
                   $total++;
                }
                echo '</tr>';
                $compteur_lignes++;
             }
      ?>
       </table>
</div>
```

On génère avec ce code, un tableau qui va contenir les emplacements des différents jours du mois.
On veille aussi à attribuer un attribut "id" à tous les éléments HTML dans lesquels on va afficher des informations dynamiques ( les liens mois suivant et mois précédent, les cellules pour les jours du mois, etc...).
Notre page est prête, nous pouvons maintenant peupler le calendrier.

### 2.2. AJAX en action avec prototype

Notre calendrier est sympathique mais désespérément vide !!
Nous allons le remplir à l'aide de prototype via une requête AJAX.
Pour se faire, nous allons donc créer un fichier que j'appellerai /js/function.js qui va se charger de générer la requête d'affichage complet du calendrier.

/js/function.js

```
tableau = function(mois,annee)
{
   var url = './ajax/ajax_calendrier.php';
   var parametres = 'mois=' + mois + '&annee=' + annee;

               var myAjax = new Ajax.Request(
                       url,
                       {
                              method: 'get',
                              parameters: parametres,
                              onComplete:
remplirCalendrier
                       }
               );
}
```

Vous ne remarquez rien ?
Fini le pavé de code où l'on déclarait un objet XMLHttpRequest en fonction des navigateurs !!
Grâce à l'objet Ajax.Request, on peut très facilement envoyer une requête à notre serveur. Faisons un petit tour des variables utilisées dans cette fonction :

- **mois et annee** : on donne ici le mois et l'année pour pouvoir générer les jours de ce mois.
- **url** : c'est l'adresse du fichier distant qui va se charger d'effectuer le traitement de calcul des jours à afficher pour le mois et l'année choisis.
- **parametres** : ce sont la liste des variables que l'on passe au script distant ajax_calendrier.php
- **myAjax** : c'est une instance d'un objet "prototype".
- **Attribut method** : la méthode d'envoi des paramètres , POST ou GET.
- **Attribut parameters** : la chaine des variables à envoyer.
- **onComplete** : fonction javascript à appeler une fois la réponse du serveur envoyée. (remplirCalendrier)

Les données de notre calendrier vont être générées grâce au fichier distant "ajax_calendrier.php" sollicité par notre fonction tableau.
Allons donc jeter un coup d'oeil sur ce fameux fichier distant.

## 2.3. Traitement distant et réponse JSON

### 2.3.1. Format du JSON

Avant toute chose, quelques mots sur JSON (JavaScript Object Notation) .
JSON est, à l'instar d'XML, un format de données. Son principal avantage est qu'il permet d'être directement traité par javascript, puisqu'il a le même format qu'un objet javascript.
De plus, prototype permet de lire ce format avec une facilité relativement déconcertante. Simple et léger, il peut convenir à de nombreuses applications.

Revenons à nos moutons. Notre fichier ajax_calendrier.php va construire un fichier JSON qui sera envoyé comme réponse à notre javascript.
A partir des paramètres mois et année, on va donc afficher le nom du mois, les liens vers les mois suivant et précédent, les jours du mois et enfin spécifier si un évènement a bien lieu pour le jour concerné.

structure de la réponse JSON
```
{
     "mois_en_cours" : "Janvier 2008"
     "lien_precedent" : "tableau('12','2007') " ,
     "lien_suivant" : "tableau('02','2008') " ,
     "calendrier": [
               {
                 "fill": "1" //(jour du mois)

               },
               {
                 "fill": "2" //(jour du mois)

               }//etc...
                  ]
}
```

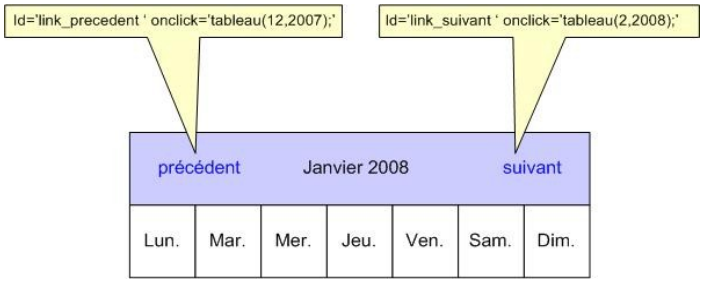
La variable fill contiendra soit :
**"vide"** si la case ne doit rien afficher (cas des lignes de début et de fin de mois).
**Le numéro du mois** (1,2, ... 31) pour la case concernée.
**Le numéro du mois sous forme de lien** si des évènements sont prévus pour le jour donné.

### 2.3.2. Calcul des dates et des évènements

Pour peupler notre calendrier, nous allons simplement construire notre réponse JSON.
Tout d'abord, on va chercher le nom du mois en fonction de son numéro, et la valeur de l'attribut onclick qu'auront les liens "mois suivant" et "mois précédent".

/ajax/ajax_calendrier.php (en tête)
```
header('Content-Type: text/x-json; charset: UTF-8');
////////////////////////////////////////////////////
////////////////////////////
//
//                    On prépare le début du retour au
format JSON
////////////////////////////////////////////////////
////////////////////////////
$retour_json='';
////////////////////////////////////////////////////
////////////////////////////
//On détermine d'abord les liens "suivant" "precedent"
et le "mois en cours" du calendrier
////////////////////////////////////////////////////
////////////////////////////
$mois=$_GET['mois'];
$annee=$_GET['annee'];
$retour_json.='{';
//mois en cours
$mois_fr=getMois($mois);
$titre=htmlentities($mois_fr." ".$annee,ENT_QUOTES);
$retour_json.='"mois_en_cours" : "'.$titre.'" , ';
//lien suivant
$date_suivant=getSuivant($mois,$annee,1);
$lien_suivant="tableau('".$date_suivant[mois]."','".$da
te_suivant[annee]."')";
$retour_json.='"lien_suivant" : "'.$lien_suivant.'" ,
';
//lien précédent
$date_precedent=getSuivant($mois,$annee,-1);
$lien_precedent="tableau('".$date_precedent[mois]."','"
.$date_precedent[annee]."')";
$retour_json.='"lien_precedent" : "'.$lien_precedent.'"
, ';
```

La réponse JSON est stockée dans la variable $retour_json.
la fonction **getMois** est une fonction permettant d'obtenir le nom du mois en français à partir de son numéro ( 1=>Janvier par exemple, ces fonctions sont définies dans le fichier /conf/func_calendrier.php disponible dans les sources).
la fonction **getSuivant** est une fonction permettant d'obtenir l'année et le mois suivants ou précédents une date donnée ( pour Janvier 2008 par exemple , getSuivant(1,2008,1) retournera le mois 2 et l'année 2008 et getSuivat(1,2008,-1) retournera 12 et 2007).
Pour comprendre le rôle variables $lien_suivant et $lien_precedent, il faut se rappeller que ses variables serviront à renseigner l'attribut onclick des liens permettant de naviguer de mois en mois.
Toujours en prenant l'exemple de Janvier 2008, on aura la page html suivante :




Nous avons notre en-tête de calendrier, déterminons maintenant les dates à afficher et les évènements associés.

/ajax/ajax_calendrier.php (corps)

```
//Maintenant, on peut peupler le calendrier sous forme
d'un tableau de 6 lignes * 7 colonnes
//On crée notre tableau de 6semaines*7jours soit 42
éléments.
//On récupère le jour qui démmarre le mois
//ON va stocker tous les jours du mois dans un tableau
tab_jours php
$tab_jours=array();
$num_jour=getFirstDay($mois,$annee);
$compteur=1;
$num_jour_courant=1;
while($compteur<43){
     if($compteur<$num_jour){
          //On gère les "blancs" du calendrier car un
mois ne commence pas toujours un Lundi.
                $tab_jours[$compteur]='';
     }else
     {
         //si la date existe, on affiche alors le jour
dans la cellule du tableau
          if(checkdate($mois,$num_jour_courant,$annee)){
               //On vérifie si un évènement a lieu ce
jour ci
               $date=$annee."/".$mois."/".$num_jour_cour
ant;
               $contenu='';
               $requete="select * from evenements where
evenement_date='".$date."'";
               $ress=mysql_query($requete);
               if($ress){
                    $nbre=mysql_num_rows($ress);

                    if($nbre>0){
                       //lien vers le script qui va
déclencher l'affichage des évènement pour le jour donné
                       $lien='<a href=\'#\'
onclick=\'showEvent(\\"'.$date.'\\");\'>'.$num_jour_cou
rant.'</a>';
                       $tab_jours[$compteur]=$lien;
                    }else
                    {

                      $tab_jours[$compteur]=$num_jour_cou
rant;
                    }
                    mysql_free_result($ress);
               }
               $num_jour_courant++;

          }else
          {
            //On gère ici les "blanc" de fin de ligne
car un mois ne se termine pas toujours un dimanche.
                    $tab_jours[$compteur]='';
          }

     }
       $compteur++;
}
```

On stocke dans le tableau $tab_jours pour les 42 cases du tableau calendrier (7 jours * 6 semaines), le contenu qui sera affiché dans la case du calendrier.
Si on est positionné sur un jour du calendrier, on regarde dans la base de données si un ou plusieurs évènements existent pour ce jour. Si c'est le cas, on stocke un lien qui permettra d'afficher les évènements (nous verrons cela un peu plus tard).
Vous remarquerez peut être l'utilisation d'une fonction getFirstDay($mois,$annee). Cette fonction se trouve dans le fichier /conf/func_calendrier.php et permet de renvoyer le numéro du premier jour d'un mois en notation française (1=>Lundi, 2=>Mardi ... 7=>dimanche).

Maintenant que l'on a toutes nos données, on peut finir le fichier JSON en parcourant le tableau $tab_jours, et on retourne cette réponse .

/ajax/ajax_calendrier.php ( renvoi du fichier JSON)

```
 ////////////////////////////////////////////////////
////////////////////////////
 // Maintenant que l'on a notre tableau d'évènements
pour chaque jour du mois
 // On finit de construire la réponse JSON
 ////////////////////////////////////////////////////
////////////////////////////
 if(!empty($tab_jours)){
    $retour_json.=' "calendrier" : [ ';
    $compteur=1;
    while ( $compteur < 43){
       if($compteur==42){
          $retour_json.=' { "fill" :
"'.$tab_jours[$compteur].'" } ';
       }else
       {
          $retour_json.=' { "fill" :
"'.$tab_jours[$compteur].'" } , ';
       }
    $compteur++;
    }
    $retour_json.=' ] ';
}
$retour_json.=' } ';
 echo $retour_json;
```

Pour chaque case de notre calendrier, on a donc défini ce qu'il faut afficher :

```
rien pour les cases vides.
un numéro de jour pour les dates qui existent.
un numéro de jour sous forme de lien pour voir les
évènements prévus ce jour-ci.
```

Notre réponse JSON est correctement formée et peut maintenant être traitée par javascript. C'est ce que nous allons voir maintenant.

### 2.3.3. Traitement de la réponse JSON.

Le flux JSON est renvoyé sous forme de variables vers notre javascript. Rappelez vous, nous avions ce code :

/js/function.js

```
            var myAjax = new Ajax.Request(
                    url,
                    {
                            method: 'get',
                            parameters: parametres,
                            onComplete:
remplirCalendrier
                    }
            );
```

L'évènement onComplete est déclenché lorsque la réponse est envoyée. Si c'est le cas, la réponse est transmise vers la fonction remplirCalendrier.
Regardons donc comment nous allons traiter ce flux JSON dans notre fonction remplirCalendrier.

/js/function.js

```
function remplirCalendrier(reponsejson) {
      //on utilise la fonction evalJSON de prototype
pour récupérer la réponse JSON
```

```
        var data=reponsejson.responseText.evalJSON();
        //On place les liens suivants,précédents et le
mois en cours
        $('link_suivant').onclick=function(){eval(data.l
ien_suivant) ;};
        $('link_precedent').onclick=function(){eval(data
.lien_precedent) ;};
        $('titre').innerHTML=data.mois_en_cours;
        //Maintenant, on affiche tous les jours du
calendrier
        var compteur=1;
        var id='';
        while(compteur<43){
           id=compteur.toString();
           $(id).innerHTML=data.calendrier[(compteur-
1)].fill;

           compteur++;
        }
}
```

Dans un premier temps, on place nos données JSON dans une variable **data**, qui, grâce à prototype sera un objet directement utilisable contenant toutes les données de notre calendrier ! (grâce à evalJSON
On accède ensuite aux valeurs via la syntaxe data.nom_de_la_variable . Dans la structure de notre JSON, nous avions spécifié que les jours du calendrier seraient stockés dans un tableau nommé "calendrier".
Pour récupérer les valeurs de ce tableau, il nous suffit alors de faire une boucle pour parcourir ce tableau et l'afficher dans la case du calendrier correspondante.

Vous remarquerez aussi la syntaxe $(element) . C'est une fonctionnalité apportée par prototype et qui équivaut au traditionnel document.getElementById(element).

## 2.4. Affichage des évènements journaliers

Notre calendrier est désormais fonctionnel et permet une navigation de mois en mois sans rechargement visible de la page.
Nous allons afficher maintenant les évènements liés aux jours de notre calendrier.

### 2.4.1. Création de la requête AJAX

Lors de la génération du flux JSON dans le fichier ajax_calendrier.php, nous avions, lorsqu'un évènement était détecté, crée le lien suivant :

/ajax/ajax_calendrier.php
```
$lien='<a href=\'#\'
onclick=\'showEvent(\\"'.$date.'\\");\'>'.$num_jour_cou
rant.'</a>';
```

L'idée est relativement simple, on va créer une fonction javascript showEvent qui va envoyer une requête AJAX vers un fichier distant php qui va aller chercher, en fonction de la date, les évènements du jour en question.

/js/function.js
```
function showEvent(date){

   var url = './ajax/ajax_commentaires.php';
   var parametres = 'date=' + date ;

            var myAjax = new Ajax.Request(
                  url,
                  {
                        method: 'get',
                        parameters: parametres,
                        onComplete:
remplirCommentaires
                  }
            );
}
```

Comme nous l'avions vu lors de notre première requête AJAX, on va utiliser l'objet prototype Ajax.request qui nous simplifie grandement la vie.

### 2.4.3. Génération de la réponse

Rien de sorcier ici. On effectue une simple requête SQL dont on extrait les données.

ajax/ajax_commentaires.php
```
<?php
     //Connexion à la base de données
        include("../conf/mysql.php");
     $date=$_GET['date'];
     $requete="select * from evenements where
evenement_date='".$date."'";
     $ress=mysql_query($requete);
     $retour='';
     if($ress){
               while($liste_evenements=mysql_fetch_asso
c($ress)){
                  $retour.=htmlentities($liste_evenemen
ts[evenement_comment],ENT_QUOTES).'<br>';
               }
               //on renvoie la réponse
               echo $retour;
     }
?>
```

### 2.4.4. Traitement et affichage de la réponse

Dans notre appel AJAX, nous avions défini que la réponse serait traitée via une fonction js remplirCommentaires, la voici :

/js/function.js
```
function remplirCommentaires(reponse){
         var commentaires=reponse.responseText;
         $('Evenements').innerHTML=commentaires;
         PullDown.panel = Rico.SlidingPanel.top(
$('outer_panel'), $('inner_panel'));
         PullDown.panel.toggle();
}
```

Comme précédemment, on récupère la réponse avec reponse.responseText.
On réinitialise le panel afin qu'il soit en position fermée, puis on le déroule.
On affiche ensuite les évènements via la syntaxe "raccourcie" prototype : $('Evenements').innerHTML=commentaires;
On crée alors simplement dans calendrier.php une balise html div ayant pour id la valeur "Evenements".

calendrier.php
```
<div id="Evenements">
</div>
```

Nous avons donc notre calendrier, fonctionnel et dynamique. Cependant, il serait plus sympathique avec quelques améliorations visuelles. C'est ce que nous allons voir maintenant avec l'aide notamment de la librairie openrico.

## 3. Aller plus loin avec OPENRICO

Jusqu'à présent, nous avons utilisé le framework prototype intégré à openrico mais pas réellement openrico lui même.
En effet, cette librairie apporte de nombreux effets graphiques javascript que l'on peut plus ou moins facilement ajouter à notre application.
Dans le cadre de notre calendrier, nous allons l'utiliser pour styliser deux affichages :

- Arrondir les bords du calendrier
- Créer un panel déroulant pour afficher les évènements

### 3.1. Arrondir les bords du calendrier

Depuis quelques temps maintenant, et inscrit dans la mouvance "2.0", on aperçoit de plus en plus de blocs html avec les bords arrondis.
Si vous êtes rompu au langage HTML, vous savez qu'il est souvent délicat de créer ce genre d'effets. OPENRICO va vous permettre d'y parvenir avec plus de facilité.
Tout d'abord, nous allons inclure les fichiers javascript nécessaires dans notre fichier calendrier.php

calendrier.php (inclusion des js openrico)
```
<script src="./js/rico/src/rico.js"
type="text/javascript"></script>
<script src="./js/rico/src/ricoStyles.js"
type="text/javascript"></script>
<script src="./js/rico/src/ricoEffects.js"
type="text/javascript"></script>
<!-- ... -->
<div id="calendrier" class="conteneur calendrier"
style="width:260px;background-color:#c6c3de;">
<!-- code du calendrier -->
</div>
</div>
<div id="Evenements">
</div>
```

Maintenant, on va appliquer le bord arrondi à la balise conteneur "conteneur calendrier" de notre application. Ce qui aura pour effet d'arrondir les bords du calendrier.
Sous l'affichage du calendrier, on va insérer le code suivant :

calendrier.php (appel de la fonction arrondir)
```
<script>
            $$('div.conteneur').each(function(e){Rico
.Corner.round(e)});
</script>
```

Pour arrondir les angles, on va simplement appliquer la fonction Rico.Corner.round qui va se charger d'arrondir les bords de tous les éléments html dont l'attribut class commence par le mot "conteneur". En gardant cette syntaxe, on observera donc les comportements suivants :

- **attribtut class="conteneur calendrier"** : les bords vont être arrondis.
- **attribtut class="conteneur test"** : les bords seront là aussi arrondis.
- **attribtut class="test"** : les bords ne seront pas arrondis.

Le calendrier arrondi, nous allons maintenant nous pencher sur l'élément HTML "Evenements" pour styliser l'affichage des évènements pour un jour donné.

### 3.2. Construire un panel déroulant

Pour utiliser les fonctions de panel déroulant dans openrico, il faut tour d'abord importer de nouvelles librairies.

calendrier.php (inclusion des js openrico)
```
<script src="./js/rico/src/ricoComponents.js"
type="text/javascript"></script>
```

Il ne nous reste plus qu'à préparer le conteneur "Evenements" pour qu'il se présente sous forme déroulante.

calendrier.php (div evements)
```
<div style="position: relative; width: 260px;">
      <div id="top-panel" style="background-color :
#9791cb;position: relative;width: 260px;z-index:
1500;">
          <a href="javascript:void(0);" id="code-
button" onclick="PullDown.panel.toggle(); return
false;">
              + Voir les évènements
          </a>
      </div>
      <div id="main-part">
        <div id="outer_panel" class="panel-top"
style="overflow: hidden; position: absolute; z-index:
1600;top: 19px; width: 260px;height: 132px;">
            <div style="position: relative;top:
1px;background-color: #c6c3de;margin:0px;border: 1px
solid #9791cb;" id="inner_panel">
                        <div id="Evenements"
style="height:150px">
                    </div>
            </div>
        </div>
      </div>
</div>
<script>
            Event.observe(window, 'load', function(){
                PullDown.panel =
Rico.SlidingPanel.top( $('outer_panel'),
$('inner_panel'));
            })
            var PullDown = {};
</script>
```

On remarque tout d'abord le lien onclick="PullDown.panel.toggle(); return false;" . C'est en cliquant sur ce lien que l'élément va dérouler.
La fonction PullDown.panel.toggle() représente en fait l'objet openrico Rico.SlidingPanel.top(element1,element2) déclaré un peu plus bas. C'est cette fonction toggle() qui va s'occuper d'enrouler / dérouler le panel selon nos besoins.
Ensuite les éléments inner_panel et outer_panel sont les éléments qui vont "bouger" lorsque l'on va cliquer sur le lien "voir les evenements".
Enfin, la div "Evenements" est la même que celle que l'on a créee tout à l'heure et servira à stocker les évènements prévus pour un jour donné.

## 4. Un petit bilan

### 4.1. Les avantages des framework

Le développement web est une pratique difficile car les contraintes sont nombreuses: multiplicité des technologies existantes, multiplicité des environnement clients (compatibilité des navigateurs ? ), besoins du client ...
L'utilisation d'un framework prend alors toute son importance ! Ces outils, si ils sont performants, vous permettent de gagner du temps en évitant de réinventer la roue, et vous garantissent généralement une réponse fiable à vos besoins.
Même si leurs documentations ne sont pas toujours claires et complètes, il est enrichissant de les pratiquer et de les mettre en oeuvre.

### 4.2. Les améliorations possibles

La mise en oeuvre de ce calendrier a avant tout un but pédagogique, et il existe bien sûr d'autres méthodologies pour obtenir un résultat similaire !
Selon vos contraintes, vous pouvez tout à fait privilégier d'effectuer bon nombre de traitements dans vos scripts javascript au lieu des scripts php distants (notamment pour l'affichage du calendrier).
De plus, vous pouvez tout à fait améliorer encore l'esthétisme de l'application en utilisant des liens images (avec des flèches par exemple) pour afficher les liens "mois suivant" et "mois précédent".
En couplant cette pratique avec une feuille de style à votre goût, vous pouvez construire une application encore plus ergonomique (en colorant les cases du calendrier qui ont des évènements par exemple)

### 4.3. Les limites

Utiliser des technologies reposant sur le client est un choix. En effet, vous devez prendre en considération le fait que des environnements utilisateurs ne vont pas prendre en charge javascript par exemple.
Essayez donc toujours de proposer une solution alternative à une fonctionnalité javascript.
De plus, vous permettrez ainsi aux moteurs de recherche de pouvoir indexer ce contenu car le javascript n'est que peu ou pas pris en compte par les "crawlers".
Toujours à propos des moteurs de recherche, pensez à interdire aux robots l'accès à vos fichiers distants php que vous utilisez avec AJAX. Cela vous évitera de voir des pages "inutiles" pollués les pages déjà indéxées.
Dans un souci de lisibilité, nous avons volontairement omis de nombreuses règles de sécurité de la programmation web, faites attention à ne pas les oublier lors de vos déploiements.

Retrouvez l'article de Julien Seignalet en ligne : Lien25

# Microsoft Volta - Partie 1 : Installation et découverte

Une des toutes dernières technologies de Microsoft nommée Volta fait de plus en plus parler d'elle. Nous allons, au cours de cet article, essayer de savoir un peu plus ce qui se cache derrière ce nom étrange.
Cet article se base sur la toute première CTP disponible au public.

## 1. Introduction

Depuis quelques temps, un nouveau projet de chez Microsoft fait de plus en plus parler de lui. Ce projet, dont la preview fut officiellement présentée sur le blog dédié (Lien26) le 5 décembre, se nomme Volta. Il s'agit d'une nouvelle technologie qui se base sur un compilateur bien spécifique puisqu'il permet de créer des applications Web à partir de code MSIL (.Net). La différence par rapport à l'ASP.Net réside dans le fait que le code fonctionnel (code behind) est transformé en JavaScript et l'application devient alors portable sur tout type de plateforme. Volta permet également mais surtout de faire du "lean programming" et de modifier l'architecture de l'application à n'importe quel moment de son développement.

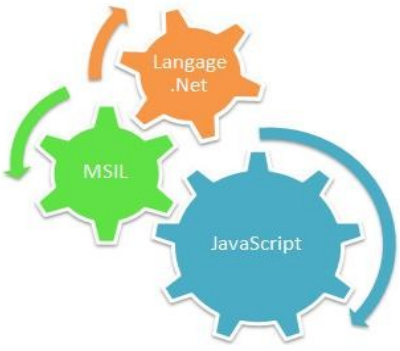
Il est très important de noter qu'il s'agit de la toute première CTP de test. Nous sommes encore loin d'une version pré-alpha. L'équipe de développement travaille pour le moment à tester la faisabilité des différentes fonctionnalités qu'ils souhaitent intégrer.

## 2. Présentation

Volta, qui n'est pour le moment qu'expérimental, a pour objectif de proposer une boîte à outils complète afin de permettre au développeur de créer rapidement des applications Web. A première vue (mais à première vue uniquement), Volta pourrait ressembler fortement à d'autres boîtes à outils tel GWT (Google Web Toolkit) et c'est justement cette ressemblance qui fit (et fera encore) parler les mauvaises langues, considérant Volta comme un simple clone de GWT.
Nous allons justement voir au sein de cet article que Volta est bien plus que cela.

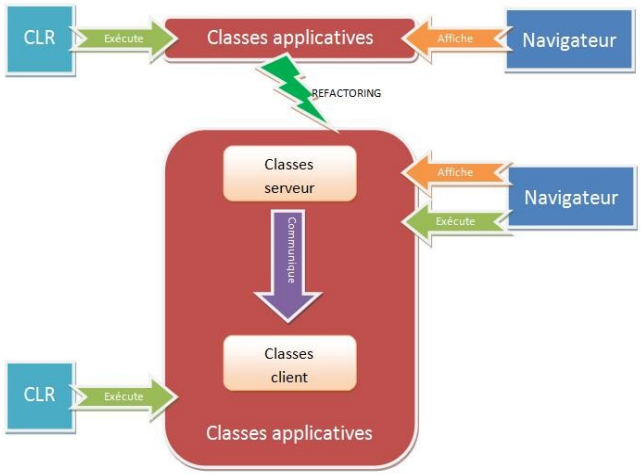
### 2.1. Fonctionnement

Volta est ce qu'on pourrait appeler un "recompilateur". En effet, il se base du code MSIL (qui comme vous le savez est obtenu après la compilation d'un langage .Net), pour générer du JavaScript tel que le montre le schéma suivant:




Cela parait "incroyable" (ou un peu "gros") dans le sens où beaucoup de gens trouvaient le JavaScript trop limité, mais l'équipe de Volta a effectivement transformé une grande partie de la plateforme .Net en équivalent JavaScript!

Ce qui parait simple sur le schéma est en fait un peu plus complexe que cela, surtout que Volta peut faire plus que de la transformation. Volta se base sur le refactoring du code, particulièrement les attributs (nous en parlerons dans la seconde partie), pour générer l'application Web souhaitée.




Comme le montre le schéma précédent, nous obtenons après compilation des classes serveurs (en .Net) et des classes clientes (celles qui ont été transformées) et déjà, par ce simple découpage, l'application est passée en application 2-tiers.
Comme nous le verrons juste après, ce découpage mais surtout cette "transformation" en JavaScript ne se contente pas de simplement essayer de reproduire le même comportement, il crée des classes entières en JavaScript capables d'agir et de réaliser leur code métier propre, tout en étant capable d'interagir avec la partie restée en .Net (si on a choisi cette solution).

### 2.2. Tiers-splitting

Un des avantages de Volta (mais que vous ne verrez que dans un second article à venir) est de pouvoir choisir comment sera découpée l'application au tout dernier moment. Cela s'appelle du "lean programming". Pour faire court et ne pas gâcher la surprise du second article, Volta permet de développer son application bien avant de se demander quelle partie sera cliente et quelle partie sera serveur.

Ainsi, on peut très rapidement lister un certain nombre d'avantages de Volta:
**Indépendance du langage** : On peut écrire du code dans n'importe quel langage qui est compilé en MSIL (C#, VB, IronPython, etc.)
**Utilisation de toute la chaîne .Net** : Utilisation des librairies, de

l'IDE, des snippets, des profiler, des outils comme FxCop ou ILDASM, etc
**Splitting** : Possibilité de découper l'application en autant de tiers que nécessaire

### 2.3. Support des navigateurs

Le but de Volta étant de générer des applications Web, il devient important que ces applications soient supportées par tout type de navigateur. Les développeurs ASP.Net 1.1 se rappellent des problèmes de normes W3C non suivies. Ici, le code est du JavaScript et n'a donc pas forcément besoin de répondre à des spécifications strictes (si ce n'est de s'approcher des normes de DHTML) mais se doit surtout de fonctionner directement sur les navigateurs courants. Ainsi donc, sans avoir rien à changer, le code compilé est garanti de fonctionner de la même façon, tant sur Internet Explorer que sur Mozilla Firefox. Volta génère effectivement des blocs de code s'adaptant au navigateur utilisé.

Nous avons donc:
**Un support multi-navigateur** : Même code pour Internet Explorer et Firefox.
**Transparence de débogage** : Debug du code avec un navigateur spécifique
**Fonctionnalités spécifiques à un navigateur** : Quand nécessaires, les fonctionnalités spécifiques d'un navigateur sont utilisées
**Intégration dans Visual Studio** : Utilisation facile via Visual Studio

## 3. Limitations

Comme toutes les bonnes choses, il y a des limitations. Si ces limitations peuvent déjà paraitre nombreuses (et méritent pour cela un chapitre entier), elles s'expliquent naturellement pour la plupart d'entre elles.
En voici la liste, telle que décrite sur le site officiel du projet.

### 3.1. Librairies

Les bibliothèques sont limitées de plusieurs façons:
- le multithreading n'est pas supporté
- La réflexion est quasi inexistante
- La BCL (Base Class Library) n'est qu'en partie supportée. Elle contient la plupart des méthodes communes aux différents langages du Framework .Net.

### 3.2. Support du langage Visual Basic

Certaines méthodes qui n'étaient alors que présentes dans Visual Basic .Net ne sont pas supportées.
Ainsi, le mot clé My et tous les services qui en découlent ne sont pas supportés
Le late-binding (au moment de l'exécution) n'est pas supporté
Les méthodes de conversion comme Cint ne marchent que si le type visé supporte la conversion
Certains éléments hérités du VB6 comme la méthode Beep ne sont pas non plus supportés.

Le non support d'éléments du VB.Net a un rapport avec le support de la BCL et les développeurs de Volta disent déjà comment savoir corriger cela pour la suite, mais ceci demandant énormément de travail, ils n'ont pas tenu à l'intégrer dans la première CTP. Cela devrait donc s'améliorer dans les futures versions de Volta.

### 3.3. Temps de téléchargement

Durant l'exécution, une application Volta télécharge plusieurs fichiers depuis le serveur, un pour chaque classe utilisée. Les classes ne sont téléchargées que lorsqu'elles sont requises, ainsi, le temps de premier lancement n'est pas affecté par le téléchargement de classes utilisées plus tard dans l'exécution. Par contre, à cause du nombre de fichiers souvent requis pour supporter l'application, les temps de chargement peuvent être affectés. Dans des versions futures, il est prévu d'ajouter une logique dans la façon de compiler et de charger les assemblies afin de répondre à cela.

Si vous vous amusez à analyser ce qui est téléchargé en lançant les samples, vous vous rendrez compte de la taille conséquente qui est téléchargée. C'est plus que conséquent pour le moment mais l'équipe de développement précise que les optimisations seront faites dans le futur.

### 3.4. Compilateur Volta

Le compilateur Volta est capable de généré du JavaScript en sortie mais ce JavaScript n'est pas entièrement optimisé. Il répondra clairement au besoin exprimé mais ne sera optimisé pour tel ou tel navigateur, ni ne fera d'optimisation en nombre de lignes de code, ceci alourdissant le poids des pages

### 3.5. Découpage et architecture

Volta qui présente comme fonctionnalité de pouvoir rapidement faire du découpage multi-tiers nécessite pour cela certains contraintes:
- Seules les sous-classes dérivant de Objet peuvent être découpées
- La sérialisation entre deux tiers n'est pas rapide
- L'URI du serveur peut seulement être spécifié en tant qu'argument d'un attribut (dans le code source). Ainsi donc, pour changer l'adresse du serveur, il faudra modifier l'argument puis recompiler le projet.
- Pour le moment, il n'est possible de ne découper l'application que sur un seul serveur

### 3.6. Imports et autres limitations

Le débogage multiple de plusieurs application Volta n'est pas possible et le binding grâce à l'attribut Import ne permet pas l'utilisation des tableaux (Arrays)

## 4. Utilisation

### 4.1. Pré-requis

Il vous faut: d'abord un environnement de développement et actuellement, c'est
- Visual Studio 2008 (beta 2 ou finale) (Version d'essai 90 jours : Lien27)
- Microsoft Volta (dernière CTP (Lien28))

Vous n'avez qu'à lancer le fichier d'installation et relance Visual Studio. Créez un nouveau projet et choisissez le framework 3.5 pour voir apparaitre les projets de type volta.

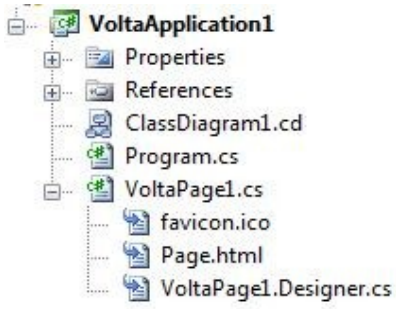
Visual Studio 2008 Express ne permettant pas de charger les extensions nécessaires à Volta, vous ne pourrez pas faire marcher Volta avec celui-ci.

### 4.2. Hello World

Commençons par l'application la plus simple qui soit, le mythique "Hello World!". Qui n'a jamais commencé par cela?
Ouvrez donc Visual Studio puis choisissez de faire un nouveau projet (Fichier > Nouveau > Projet) puis choisissez un projet de type Volta Application (le filtre en haut à droite doit être placé sur

Framework 3.5)

Vous remarquez alors en regardant l'explorateur de solution qu'il contient un élément VoltaPage1, fichier class C#, lui même relié à un designer de cette même classe mais également à un fichier Page.html qui sera la partie visuelle de votre application.
Pour ce premier exemple, nous ne nous occuperons pas de cet élément bien qu'il soit utilisé une fois la compilation accomplie.




Dans le projet vidé crée, vous constaterez que votre classe hérite de Page, mais pas l'objet Page que vous utiliseriez en ASP.Net, non je parle de l'objet Microsoft.LiveLabs.Volta.Html.Page. Celui-ci ne marche pas exactement de la même façon mais vous en rendrez compte au fur et à mesure que vous découvrirez Volta.
Continuons donc et remarquons notre constructeur. Même pas de Page_Load, juste un constructeur.
Vous remarquerez aussi les deux using qui nous intéressent:

Imports principaux

```
using Microsoft.LiveLabs.Volta.Html;
using Microsoft.LiveLabs.Volta.Xml;
```

Modifions maintenant notre constructeur dans lequel nous allons rajouter quelques lignes:

Hello World

```
Button b = new Button();
b.Value = "Hello?";
b.Click += delegate { Window.Alert("Hello
Developpez.com!"); };
Document.Body.AppendChild(b);
```

Nous créons un bouton, lui spécifions son texte via la propriété Value (comme en HTML), puis nous lui assignons un évènement (comme en C#) mais dans lequel nous tapons directement notre code JavaScript. Enfin nous ajoutons notre composant bouton à notre page. Compilons et exécutons.

Une page html est alors lancée et...magie, nous y retrouvons notre bouton. Cliquons alors dessus pour voir un message "Hello Developpez!" s'afficher. Notre Hello World fonctionne!

Analysons ce qui s'est réellement passé. Nous avions du code C# qui au final nous génère plusieurs fichiers. Nous avons notre fichier Page.html avec un squelette de page vide mais respectant les standards Web!, des fichiers JavaScript génériques VoltaManagedInteropHelpers.js et compact.js (pour la compatibilité avec d'autres navigateurs. Mais sont également générés des fichiers pour représenter l'assembly de l'application (assembly.js) et des objets en JavaScript représentant notre couche métier. Ainsi donc, notre bouton et sa méthode, que l'on a écrit en 4 lignes se retrouve transcrit en 13 grosses lignes de code

C'est à ce moment que l'on se rend compte d'où vient le poids des pages Volta. Je vous laisse déjà imaginer le poids avec une application plus complète.

### 4.3. Release

"Release", pourquoi nous parle-t-il de release. Et bien essayez vous même et compilez votre projet en mode Release puis analysez le code généré.
- La page.html a été modifiée et contient directement le code qui appelle les objets Javascript
- Un fichier jsfile.aspx est crée et qui contient les entêtes normaux d'une page aspx
- Un fichier loader.js est crée, qui lui, va se charger de charger tous les éléments nécessaires et se servir de la page aspx comme noyau central de chargement, en lui passant des paramètres.

C'est, je trouve, uniquement en mode release que l'on peut réellement se rendre compte du fonctionnement de Volta même s'il faut pour cela s'amuser à lire et analyser les centaines de lignes de code JavaScript générées.

### 4.4. Cas réel

Un Hello World c'est bien joli, mais ce n'est pas avec cela qu'on fait une application. D'ailleurs ce n'est pas avec l'exemple suivant non plus que vous ferez une application néanmoins, je pense qu'il est intéressant pour commencer à voir comment il est possible de coder comme nous le ferions en ASP.Net (mais aussi en JavaScript+DOM) pour arriver à créer simplement l'application qui sera portable sur n'importe quelle plateforme.
Nous allons implémenter un formulaire de login tout ce qu'il y a de plus simple, qui affichage un message en cas d'erreur et qui redirige vers la partie privée (ici Developpez.com, lorsque les credentials sont identiques)

Recréons un nouveau projet de type Volta Application, que nous nommerons VoltaApplication2 (10 minutes pour trouver ce nom très recherché! :))

Intéressons nous cette fois au fichier page.html dans lequel nous allons dessiner notre code xHTML contenant un bouton et deux textbox mais aucun code, pas même une balise <form>

```
<body>
  <table id="frmLogin">
    <tr>
      <td>
        Login :</td>
      <td>
    <input id="txtLogin" type="text" /></td>
    </tr>
    <tr>
      <td>
        Mot de passe :</td>
      <td>
    <input id="txtPwd" type="password" /></td>
    </tr>
    <tr>
      <td colspan="2">
        <input id="btnLogin" type="button" value="Login"
/></td>
    </tr>
  </table>
  <div id="msgBox"></div>
```

```
</body>
```

Vous avez remarqué les ID placés sur les contrôles. Qu'importe le langage, c'est le seul moyen de viser directement tel ou tel contrôle. Vous aurez aussi remarqué qu'il y a pas de tag runat="server" comme vous auriez en ASP.Net.
Mais alors, comme cela va-t-il marcher? Et bien, nous allons utiliser DOM à travers du code C# tout simplement:

Dans notre constructeur, nous n'allons plus créer un bouton mais récupérer celui que nous avons placé dans le code HTML, en utilisant la méthode bien connue GetById. Ici, nous n'utilisons pas directement la méthode JavaScript qui fait cela, mais une classe wrapper qui le fait pour nous et qui a été créée par l'équipe de Volta.
Puis nous ajoutons un événement mais cette fois, nous appelons une autre méthode C#, et pas une fonction JavaScript comme dans le Hello World.

Constructeur

```
Input btn = Document.GetById<Input>("btnLogin");
btn.Click += btnLogin_Click;
```

Regardons maintenant ce que nous allons mettre dans notre méthode btnLogin_Click.Voici son code:

Méthode btnLogin_Click

```
void btnLogin_Click(){
   Input txtLogin = Document.GetById<Input>("txtLogin");
   Input txtPwd = Document.GetById<Input>("txtPwd");
   if ( txtLogin.Value != String.Empty && txtLogin.Value ==
txtPwd.Value){
      Document.GetById<HtmlElement>("msgBox").InnerHtml =
"<font color='green'>Bienvenue!</font>";
      Document.GetById<HtmlElement>("frmLogin").Style.Displ
ay = "none" ;
      this.Window.Alert("Maintenant la redirection");
      this.Window.Navigate("http://www.developpez.com");
   }
   else Document.GetById<HtmlElement>("msgBox").InnerHtml
= "<font color='red'>Try again!</font>";
}
```

Sur les deux premières lignes, nous récupérons une référence vers nos objets HTML. Puis nous testons si les valeurs sont identiques et agissons en fonction du test.
Pour les développeurs ASP.Net, cela ne les change pas de leurs habitudes même si je vous rappel que nos contrôles n'ont pas été mis en tant que contrôles serveurs pour pouvoir y accéder.
Sur les lignes suivantes, j'affiche un message en vert et je cache le formulaire de login lorsque le login est équivalent au mot de passe.

Notez bien par contre, les deux lignes suivantes:

```
this.Window.Alert("Maintenant la redirection");
this.Window.Navigate("http://www.developpez.com");
```

Vous vous doutez de ce que cela fait mais néanmoins, cela ne fait pas partie du framework .Net. Là encore, il s'agit de méthodes contenues dans une classe wrapper qui vous permet très simplement d'agir comme du JavaScript. Fini les ClientRegisterStartupScript, grâce aux très nombreuses classes wrapper, vous pouvez écrire du code côté client très simplement.

### 5. Conclusion

C'en est fini pour cette présentation mais vous aurez rapidement un second article qui vous montrera plus en détails les avantages principaux de Volta. En attendant, nous avons déjà pu nous rendre compte d'un certain nombre de choses concernant Volta.
Dans un article à venir, nous verrons le découpage multi-tiers et le fonctionnement des attributs mais nous verrons également comment utiliser AJAX au sein de nos applications Volta.

Retrouvez l'article de Louis-Guillaume Morand en ligne : Lien29

# Les livres .Net

## Programming WPF, Second Edition

### Critique du livre par Benjamin Roux

Comme annoncé dans la préface par les auteurs, ce livre s'adresse aux développeurs, et non aux designers, mais plus particulièrement aux développeurs initiés qui ont déjà une certaine pratique du C# ou de la plateforme .NET en général. Aucun rappel n'est effectué sur les bases du C# et c'est tant mieux, il en reste plus pour le vif du sujet.

Parlons en d'ailleurs, dès le premier chapitre on rentre dedans, avec notre premier Hello en WPF. Le reste de ce chapitre est tout simplement une sorte d'introduction aux futurs chapitres avec des petits exemples (comme pour le Data Binding) ou sous forme d'image (comme pour la 3D).

Les chapitres suivants passent en revue les principales caractéristiques de WPF, avec par un exemple un chapitre sur la liste des contrôles communs (Bouton, TextBox...), un autre sur les différents positionnements des contrôles dans les fenêtres (StackPanel, Grid...), mais aussi par le très intéressant et selon moi, le plus intéressant, Data Binding.
On retrouvera également des chapitres sur les très intéressants, Animations, et 3D Graphics, pour faire des applications WPF jolies.

Le reste des chapitres vous expliquera les rudiments de WPF, comme la navigation (applications XBAPs), la création et l'application de styles sur vos contrôles, les Graphics (images...) ou même mettre du texte en forme, de mille et une façons.

Le tout est toujours agrémenté d'images pour facilité la compréhension, ainsi que de codes sources expliqués pour une meilleure compréhension.
On saluera également la présence d'appendices avec entre autres le très intéressant Silverlight.

En conclusion, ce livre va je pense devenir Le livre de référence sur WPF, tout comme le sont la plupart des livres aux éditions O'Reilly dans leur catégorie.

Retrouvez cette critique de livre sur la page livres .NET : Lien30

# Les derniers tutoriels et articles

## Développer une application Oracle en C/C++ avec la librairie OCILIB

Comment programmer simplement et efficacement des applications Oracle performantes en C/C++.

### 1. Présentation

#### 1.1. Introduction

OCILIB est une librairie Open Source multi plateforme, écrite en C, proposant une gamme d'APIs permettant :

- D'accéder très facilement à des bases de données Oracle.
- D'exécuter des ordres SQL et PL/SQL
- Manipuler les données d'une base de données

#### 1.2. Complexité d'OCI

Oracle Call Interface (OCI) (Lien31) est une API fournie par Oracle permettant aux développeurs de créer des applications en utilisant des appels C/C++ bas niveau afin d'accéder à des bases de données Oracle. OCI permet de contrôler tous les aspects de l'exécution d'ordres SQL tout en supportant les types de données, les conventions d'appel, la syntaxe et la sémantique des langages C, C++.

OCI est une API très puissante (c'est l'API la plus bas niveau fournie par Oracle) et utilisée par beaucoup d'applications et outils (à commencer par Sql*Plus) mais aussi très complexe et lourde à utiliser...

Par exemple, se connecter à Oracle requiert au minimum quasiment 100 lignes de code ! De plus, la plupart des fonctions d'OCI requiert souvent une liste d'arguments assez conséquente (souvent une dizaine de paramètres, voir plus...)

Au final, une application OCI est souvent pénible à coder, relire et maintenir et les "experts" en OCI ne sont pas aussi nombreux que les codeurs Java, .NET ou VB !

#### 1.4. Pourquoi OCILIB ?

OCILIB encapsule OCI afin de fournir une interface beaucoup plus simple à coder, lire et surtout afin de permettre une réutilisabilité du code optimale.

OCILIB est gratuit (open source - license LGPL) et fournit plus de 270 fonctions simples et son code source est indépendant de toute plateforme.

Les langages C et C++ sont parmi les plus performants et malgré cela, souvent lors du choix d'un langage pour coder une application Oracle, ils ne sont pas retenus, au profit de frameworks de type JAVA et .NET. Ils sont écartés bien souvent pour des raisons d'accès car les API C/C++ pour accéder à Oracle (fournies ou non par Oracle) sont complexes à mettre en oeuvre.

OCILIB permet de concilier les performances de C/C++ avec une mise en oeuvre très simple via une interface la plus pragmatique possible tout en gardant une richesse dans les fonctionnalités.

OCILIB est une des rares librairies C/C++ basées sur OCI apportant un support complet Unicode. Une application OCILIB peut être indifféremment compilée nativement en Ansi ou en Unicode.

Enfin, le code d'OCILIB est 100 % ISO C (C89 et C99 pour l'Unicode) ce qui lui permet d'être extrêmement portable !

OCILIB ne nécessite pas de client Oracle pour développer. En effet, une application utilisant OCILIB peut linker les librairies Oracle à la compilation (liaison statique ou dynamique) ou charger dynamiquement ces librairies en runtime si l'OS cible supporte le chargement dynamique de modules.

OCILIB fonctionne sur toutes les plateformes supportées par Oracle (Microsoft Windows, Linux, Unix, ...).

### 2. Installation

#### 2.1. Prérequis

Posséder un système Unix like ou Microsoft Windows supporté par Oracle
Avoir un client Oracle installé (non requis pour la compilation mais pour l'exécution)

#### 2.2. Compatibilités

Les plateformes suivantes ont été validées (compilation, installation, build et exécution d'un projet) :

- Microsoft Windows
- Linux
- Solaris
- HP/UX
- AIX

Les autres plateformes supportées par Oracle (MacOS server, OpenVMS, z/OS) n'ont pas été officellement testées, mais au vu des plateformes déja validées, cela ne devrait pas être trop problématique !

Les compilateurs suivants ont été validés :

Microsoft Compilers (VC++, VS200X)

- GCC et MinGW
- XLC
- CC propriétaires

Les versions d'Oracle suivantes (clients et serveurs) ont été validées :

- Oracle 8i
- Oracle 9i
- Oracle 10g

- Oracle 11g

### 2.3. Installation sous Unixes (Unix/Linux/Mac)

OCILIB utilise les outils GNU pour son déploiement et son installation.

- Décompresser l'archive courante (ocilib-x.y.z-gnu.tar.gz)
- $ cd ocilib-x.y.z
- $ ./configure
- $ ./make
- $ ./make install (généralement, il faut passer en root pour l'installation)

Vérifier la variable d'environnement des chemins d'accès des librairies dynamiques (LD_LIBRARY_PATH, LD_PATH, ..) afin que le répertoire des shared libraries d'Oracle s'y trouve ainsi que celui où est installé OCILIB (typiquement /usr/local/lib sous linux)

Typiquement, il suffit de rajouter à son fichier .profile :

> export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib:/usr/local/lib

OCILIB supporte 3 options supplémentaires d'installation :

- --with-oracle-import=(linkage|runtime)
- --with-oracle-charset=(ansi|unicode|mixed)
- --with-oracle-home=(oracle directory)

### 2.4. Installation sous Microsoft Windows

Sous Microsoft Windows, des DLLs sont fournies (elles peuvent être recompilées aisément)

- Décompresser l'archive courante (ocilib-x.y.z-win32.zip)
- Copier ocilib/inc/ocilib.h dans un répertoire inclus dans la liste de l'option "fichier headers" du compilateur
- Copier ocilib/inc/ocilib[x].lib dans un répertoire inclus dans la liste de l'option "fichier bibliothèques" de l'éditeur de lien
- Copier ocilib/inc/ocilib[x].dll dans un répertoire inclus dans la variable d'environnement PATH

[x] représente la version compilée d'OCILIB : "a" pour Ansi, "u" pour Unicode et "m" pour mixed mode

### 2.5. Configuration de projets

Pour utiliser OCILIB dans un projet, il faut inclure le header "ocilib.h".

Certaines options doivent être définies avant l'inclusion du header ocilib.h

=> Mode de liaison Oracle :

- OCI_IMPORT_LINKAGE : pour lier les librairies (static or shared) à la compilation (option par défaut sous Unix like)
- OCI_IMPORT_RUNTIME : pour charger dynamiquement les librairies oracle à l'exécution (option par défault sous MS Windows)

=> Charsets utilisés :

- OCI_CHARSET_ANSI : toute les chaînes de caractères sont en Ansi (option par défaut)
- OCI_CHARSET_UNICODE : toute les chaînes de caractères sont en Unicode (versions de Oracle >= 9i)
- OCI_CHARSET_MIXED : Ordres SQL, métadatas en Ansi et données utilisateurs fournies et récupérées des requêtes en Unicode

=> Convention d'appel (MS Windows uniquement) :

- OCI_API : non défini (option par défaut)
- OCI_API : __sdtcall pour utiliser les dll précompilées

Sous Windows, pour utiliser OCILIB en ANSI, en utilisant les Dlls fournies, il suffit de créer un nouveau projet et :

- Inclure "ocilib.h"
- Définir OCI_API=__sdtcall dans les options du préprocesseur du projet
- Si toutes les variantes ocilib[x].lib sont disponibles pour le linker, il faut alors préciser quel version en insérant #pragma comment(lib, "ocilib[x].lib") dans un des fichiers du projet

Sous Linux, pour utiliser OCILIB en ANSI et avec un linkage des shared librairies à la compilation, il faut :

=>Inclure "ocilib.h"

=> Ajouter au makefile pour le compilo :

- -I/$ORACLE_HOME/rdbms/public pour inclure les headers Oracle
- -I/usr/local/include pour le header ocilib.h
- -DOCI_IMPORT_LINKAGE -DOCI_CHARSET_ANSI pour configurer ocilib

=> Ajouter au makefile pour le linker :

- -L/$ORACLE_HOME/lib -lclntsh pour les librairies Oracle
- -L/usr/local/lib -locilib pour la librairie OCILIB

## 3. Se Connecter à Oracle

### 3.1. Initialiser OCILIB

Avant tout chose, OCILIB doit être initialisée. Pour cela il faut, avant tout appel à une fonction de la librairie, appeler OCI_Initilalize(). Cette fonction initialise la librairie et prend en paramètres :

- [Optionnel] Une pointeur sur une fonction de gestion des erreurs
- [Optionnel] Un répertoire ou se trouve les librairies Oracle (si runtime loading et plusieurs clients Oracle installés)
- [Optionnel] Le mode d'initialisation. Actuellement, seule la valeur OCI_ENV_DEFAULT est supportée

Exemple

```
#include "ocilib.h"

int main()
{
    if (OCI_Initialize(err_handler, NULL,
OCI_ENV_DEFAULT) == FALSE)
        return EXIT_FAILURE;

    /* ... code application ... */

    OCI_Cleanup();

    return EXIT_SUCCESS;
}
```

En fin d'application, un appel à OCI_Cleanup() est nécessaire pour :

- Désallouer les objets non explicitement libérés
- Décharger les librairies Oracle dans le cas d'un runtime loading

### 3.2. Connexions

Se connecter à une base de données Oracle s'effectue par la fonction OCI_CreateConnection() qui prend en paramètres :

- Le nom du service Oracle (alias Oracle, SID, ... en fonction des paramètres du client Oracle).
- Le nom du user Oracle sur lequel se connecter
- Le mot de passe du user Oracle
- Le mode de session (OCI_SESSION_DEFAULT, OCI_SESSION_SYSDBA, OCI_SESSION_SYSOPER)

Cette fonction a la charge de se connecter au serveur, créer une session et d'établir une transaction. En cas de succès, elle renvoie un handle de connexion et l'application peut de suite exécuter des ordres SQL.

Exemple

```
    OCI_Connection *cn;

    /* .... */

    cn = OCI_CreateConnection("Database", "user",
"password", OCI_SESSION_DEFAULT);

    if (cn != NULL)
    {
        printf("%s\n", OCI_GetVersionServer(cn));

        /* ... code application ... */

        OCI_FreeConnection(cn);
    }

    /* .... */
```

Pour fermer la connexion au serveur, il suffit d'appeler OCI_FreeConnection()

### 3.3. Gestion des erreurs

OCILIB propose un mécanisme de gestion des erreurs basé sur la notion de callback qui sera déclenché dans les cas suivants :

- Erreur générée par OCI
- Erreur d'allocation mémoire
- Erreur internes à OCILIB

Le prototype de la fonction à fournir est le suivant :

```
typedef void (*POCI_ERROR) (OCI_Error *err);
```

Un handle sur un object OCI_Error est alors fourni à la fonction. La librairie fournit des fonctions pour accéder aux propriétés de l'erreur générée.

Exemple d'une fonction de gestion des erreurs qui se contente d'afficher à l'écran une erreur SQL :

```
void ErrorHandler(OCI_Error *err)
{
    int sql_code = OCI_ErrorGetOCICode(err);

    if (sql_code != 0)
    {
        printf( "Code : ORA-%05i\n"
                "Msg : %s\n",
                sql_code, OCI_ErrorGetString(err));
    }
}
```

### 3.4. Transactions

OCILIB supporte les mécanismes de transactions (locales et globales) proposées par Oracle via OCI.

Par défaut, une fois que l'application s'est connectée à Oracle, une transaction locale est créée.
Toute modification de données (insert, update, delete) n'est pas visible des autres sessions tant que l'on a pas explicitement validé les modifications par un appel à OCI_Commit().

Si l'application ne veut pas valider ses modifications et donc annuler les modifications de données effectuées depuis la dernière validation (ou par défaut depuis la connexion au serveur), elle doit utiliser OCI_Rollback().

Il y des cas ou les simples transactions locales par défaut ne suffisent pas :

- gérer une connexion en lecture seule par exemple
- gérer une transaction globale dans un environnement distribué
- etc,..

Dans ces cas là, OCILIB propose une gestion explicite des transactions où il est possible de créer une transaction et de l'associer à un handle de connexion.

### 3.5. Types de donnée supportés

OCILIB supporte tous les types de données fournis par Oracle, à l'exception des références et collections.

Liste des types supportés :

- Tous les types scalaires (strings, numériques, flottants, ..) : CHAR/NCHAR, VARCHAR2/NVARCHAR2, NUMBER, FLOAT, ...
- Types binaires : RAW, LONG RAW, VARRAW, ..
- Larges Objects (Lobs et Files) : BLOB, CLOB, NCLOB, BFILE, CFILE
- Types LONGs : LONG, VAR LONG, LONG RAW, ...
- Date, Timestamps et Intervals : DATE, TIMESTAMP (tous), INTERVAL (tous)
- PL/SQL types : Ref cursors et Nested Tables
- Names Types : User types, XmlType, ....
- ROWIDs

Tous ces types supportés peuvent être liés à des statements ou être récupérés d'un select.

| TYPE Oracle | Type OCILIB |
|---|---|
| Strings : CHAR/NCHAR, VARCHAR2/NVARCHAR2, .... | dtext * (char * or wchar_t * selon le build) |
| Numbers sans précision (entiers) : INT, NUMBER, ... | int |
| Flottants : FLOAT, REAL, NUMBER(X,Y), .... | double |
| RAW | void * |

| | |
|---|---|
| LONG, LONG RAW, VARRAW, .. | OCI_Long |
| BLOB / CLOB / NCLOB | OCI_Lob |
| DATE | OCI_Date |
| TIMESTAMP, TIMESTAMP_TZ, TIMESTAMP_LTZ | OCI_Timestamp |
| INTERVAL, INTERVAL_YM, INTERVAL_DS | OCI_Interval |
| PL/SQL Ref Cursors | OCI_Statement |
| PL/SQL Nested Tables | OCI_Statement |
| User Types | OCI_Object |
| ROWID | dtext * (char * or wchar_t * selon le build) |

## 4. Exécuter des ordres SQL

### 4.1. SQL Statements

Afin d'exécuter des ordres SQL sur la base de données, il faut créer un object SQL statement via la fonction OCI_CreateStatement().

Un fois cet objet crée, il est possible d'exécuter des requêtes par la fonction OCI_ExecuteStmt().

Exemple

```
    OCI_Connection *cn;
    OCI_Statment *st;

    /* ... */

    st = OCI_CreateStatement(cn);

    OCI_ExecuteStmt(st, "delete from my_table where
code is null");

    printf("%d row deleted", OCI_GetAffectedRows(st));

    OCI_FreeStatement(st);

    /* ... */
```

OCI_ExecuteStmt() a donc préparé et exécuté l'ordre SQL et OCI_GetAffectedRows() a récupéré le nombre de lignes supprimées dans la table.

Une fois que le statement n'est plus utile, un appel à OCI_FreeStatement() libère toutes les ressources associées (resultset, ...).

Une object OCI_Statement peut être réutilisé pour exécuter autant d'ordres SQL que nécessaire.

### 4.2. Lier des variables

Dans l'exemple précédent, la requête était simple et ne nécessitait aucune variable d'entrée. Souvent, il est nécessaire de fournir à une requête des valeurs d'entrées non connues à l'avance. De plus, il est utile de pouvoir exécuter plusieurs fois un même ordre SQL avec des valeurs différentes sans pour autant avoir à faire repréparer le SQL par Oracle afin d'optimiser les performances.

Il est donc possible de lier des variables du programme en fournissant leur adresse au statement.

Dans ce cas, il faut :

- Préparer le SQL par OCI_Prepare()
- Lier les variables avec les fonctions OCI_bindXXX() ou XXX est le type de donnée
- Exécuter le SQL par OCI_Execute()

Exemple

```
    OCI_Connection *cn;
    OCI_Statment *st;
    int code;

    /* ... */

    st = OCI_CreateStatement(cn);

    OCI_Prepare(st, "delete from test_fetch where code
= :code");
    OCI_BindInt(st, ":code", \&code);

    code = 3;
    OCI_Execute(st);
    printf("%d row deleted"; OCI_GetAffectedRows(st));

    code = 56;
    OCI_Execute(st);
    printf("%d row deleted"; OCI_GetAffectedRows(st));

    /* ... */
```

Pour faire une insertion de masse, on peut procéder de la manière suivante :

Exemple

```
    OCI_Connection *cn;
    OCI_Statment *st;
    int code;
    char name[30];
    char value[20];

    /* ... */

    st = OCI_CreateStatement(cn);

    OCI_Prepare(st, "insert into my_table values(:code,
:name, :value)");
    OCI_BindInt(st, ":code", \&code);
    OCI_BindString(st, ":name", name, 30);
    OCI_BindString(st, ":value", value, 20);

    for (code = 1; code < 10000; code++);
    {
        sprintf(name, "name %i", code);
        sprintf(value, "value %i", code);

        OCI_Execute(st);
    }

    /* ... */
```

Néanmoins, Oracle supporte les "bulks operations" (cf. chapitre VI) qui permettent de manipuler des tableaux de variables et donnent des performances incomparables.

### 4.3. Récupérer le résultat d'un select

Récupérer le résultat d'une requête est très simple. Il suffit une fois le statement exécuté via OCI_Execute() ou OCI_ExecuteStmt() d'appeler la fonction OCI_GetResultSet() qui

retourne un handle sur un objet OCI_Resultset.

Ce resultset comprend 2 choses :

```
La description des colonnes retournées par la requête
Les données (lignes de résultats)
```

Afin d'accéder aux valeurs de chaque colonne du resultset, il faut:

Parcourir le resultset par un appel à OCI_FetchNext() qui retourne TRUE tant qu'il y a des lignes de résultat
Appeler une des fonctions OCI_GetXXX() en précisant l'index de la colonne (ou son nom) et où XXX est le type : Int, String, Date, ...

Exemple

```
    OCI_Connection *cn;
    OCI_Statment *st;
    OCI_Resultset *rs;

    /* ... */

    st = OCI_CreateStatement(cn);

    OCI_ExecuteStmt(st, "select * from test_fetch");

    rs = OCI_GetResultset(st);

    while (OCI_FetchNext(rs))
    {
        printf("code: %i, action : %s, price : %g, date
%s\n",
                OCI_GetInt(rs, 1) , OCI_GetString(rs,
2),
                OCI_GetDouble(rs,3),
OCI_GetString(rs,4));
    }

    printf("\n%d row(s) fetched\n",
OCI_GetRowCount(rs));

    /* .... */
```

OCI_GetRowCount() donne le nombre de lignes du resultset déjà parcourues.

Le même exemple avec un accès aux colonnes par leur nom :

Exemple

```
    OCI_Connection *cn;
    OCI_Statment *st;
    OCI_Resultset *rs;

    /* ... */

    st = OCI_CreateStatement(cn);

    OCI_ExecuteStmt(st, "select * from test_fetch");

    rs = OCI_GetResultset(st);

    while (OCI_FetchNext(rs))
    {
        printf("code: %i, action : %s, price : %g, date
%s\n",
        OCI_GetInt2(rs, "CODE") , OCI_GetStrin2g(rs,
"ACTION"),
        OCI_GetDouble2(rs,"PRICE"),
OCI_GetString2(rs,"DATE"));
    }

    printf("\n%d row(s) fetched\n",
OCI_GetRowCount(rs));

    /* .... */
```

Un objet statement peut être réutilisé autant de fois que voulu. De plus, il est possible d'en créer plusieurs et de les imbriquer.

Exemple

```
    OCI_Connection *cn;
    OCI_Statment *st1, *st2;
    OCI_Resultset *rs;

    /* ... */

    st1 = OCI_CreateStatement(cn);
    st2 = OCI_CreateStatement(cn);

    int code;

    OCI_ExecuteStmt(st1, "select code from my_table");
    OCI_Prepare(st2, "delete from my_table2 where
code_ref = :code");
    OCI_Bind(st2, ":code", &code);

    rs = OCI_GetResultset(st);

    while (OCI_FetchNext(rs))
    {
        code = OCI_GetInt(rs, 1);

        OCI_Execute(st2);
    }

    /* ... */
```

Il est possible de récupérer les informations relatives (metadatas) à chaque colonne du resulset (colonnes sélectionnées par la requête). Pour cela :

Un appel à OCI_GetColumnCount() permet de savoir le nombre de colonnes contenues dans le resultset
Un appel à OCI_GetColumn() en précisant l'index (position dans le select) permet de récupérer un handle sur un objet OCI_Column
Toute les propriétés de la colonne sont accessibles par une série de fonctions du type OCI_GetColumnXXX() où XXX est la propriété

Exemple

```
    OCI_Connection *cn;
    OCI_Statment *st;
    OCI_Resultset *rs;
    OCI_Column *col;

    /* ... */

    st = OCI_CreateStatement(cn);

    OCI_ExecuteStmt(st, "select * from my_table");

    rs = OCI_GetResultset(st);
    nb = OCI_GetColumnCount(rs);

    for(i = 1; i <= nb; i++)
    {
        col = OCI_GetColumn(rs, i);

        printf("%-20s%-10s%-8i%-8i%-8i%-s\n",
```

```
                OCI_GetColumnName(col),
                OCI_GetColumnSQLType(col),
                OCI_GetColumnSize(col),
                OCI_GetColumnPrecision(col),
                OCI_GetColumnScale(col),
                OCI_GetColumnNullable(col) == TRUE ?
"Y" : "N");
    }

    /* ... */
```

### 4.4. PL/SQL blocks

Des variables peuvent être liées en entrée comme en sortie.

Pour exécuter du PL/SQL, il suffit d'englober le code PL/SQL par un "begin" en début et un "end;" en fin de block.

Exemple d'un bloc PL/SQL

```
    OCI_Connection *cn;
    OCI_Statment *st;

    /* ... */

    st = OCI_CreateStatement(cn);

    OCI_Prepare(st, "begin :n := trunc(sysdate+1)-
trunc(sysdate-1); end;");
    OCI_BindInt(st, ":n", &n);
    OCI_Execute(st);

    printf("Result : %i\n", n);

    /* ... */
```

Exemple de récupération d"un cursor PL/SQL et fetch de son résultat dans le programme C :

Exemple

```
    OCI_Connection *cn;
    OCI_Statment *st;

    /* ... */

    st = OCI_CreateStatement(cn);
    st2 = OCI_CreateStatement(cn);

    OCI_Prepare(st, "begin open :c for select * from
my_table; end;");
    OCI_BindStatement(st, ":c", st2);
    OCI_Execute(st);

    rs = OCI_GetResultset(st2);

    while (OCI_FetchNext(rs))
    {
        printf("mon nom est %s\n", OCI_GetString(rs,
1));
    }

    /* ... */
```

### 4.5. Support de la clause SQL RETURNING

Oracle propose une fonctionnalité très intéressante qui est la clause SQL "returning into".

L'utilisation de cette clause dans un DML (insert/update/delete) permet de combiner 2 requêtes en 1, réduisant ainsi le nombre d'allers-retours entre le client et le serveur.

L'ajout de cette clause permet de sélectionner les lignes affectées par le DML au sein de la même requête. Ce qui est pratique, notamment dans le cas de delete car les valeurs supprimées sont ainsi récupérables !

OCILIB implémente cette fonctionnalité en créant un objet OCI_Resultset à partir des colonnes sélectionnées dans la clause RETURNING. Ce resultset est ensuite manipulable comme un resultset classique.
L'Array Interface est également compatible avec cette clause. Dans ce cas, chaque entrée du tableau peut affecter différentes lignes. OCILIB crée alors un resultset pour chaque entrée du tableau.
Par exemple, si le tableau utilisé pour lier des variables au statement contient 100 éléments, l'application commence par récupérer le premier resulset par un appel à OCI_GetResulset(). Une fois ce resultset parcouru, le resultset suivant est récupérable par un appel à OCI_GetNextResultset().

OCILIB ne pré-parse jamais les requêtes SQL (pour des raisons de performances) et donc ne peut déterminer les champs (ainsi que leur type) utilisés dans la clause RETURNING.
Il faut donc que l'application indique explicitement les colonnes de la clause RETURNING par les fonctions OCI_RegisterXXX() ou XXX est le type de données. OCILIB construit alors les resultsets sur la base des colonnes ainsi déclarées.

Exemple de sélection des objets d'une table :

```
            OCI_Connection *cn;
            OCI_Statement *st;
            OCI_Resultset *rs;

            /* ... */

            st  = OCI_CreateStatement(cn);

            OCI_Prepare(st, "update my_table set content
= content || to_char(code) returning code, content into
:i, :s");
            OCI_RegisterInt(st, ":i");
            OCI_RegisterString(st, ":s", 50);

            /* mise à jour des données */

            OCI_Execute(st);

            /* fetch des données modifiées */

            rs = OCI_GetResultset(st);

            while (OCI_FetchNext(rs))
            {
                printf("code : %i - content : %s\n",
OCI_GetInt(rs, 1), CI_GetInt(rs, 2));
            }

            printf("count : %i\n", OCI_GetRowCount(rs));

            OCI_Commit(cn);

            /* ... */
```

### 4.6. Contrôler les statements

OCILIB propose une série de fonctions permettant de personnaliser les comportements d'un statement.

Par exemple, il est possible de spécifier :

```
Le nombre de lignes pré-fetchées par le client Oracle
```

```
afin de diminuer le nombre de roundtrips avec le
serveur
Le nombre de lignes fetchées internalement par OCILIB
afin de réduire les appels OCI
Le mode de liaison des variables : par position ou par
nom
Le format de date par défaut
Etc, ..
```

Retrouvez la suite de l'article de Vincent Rogier en ligne : Lien32

## Nokia s'offre Trolltech pour la nouvelle année

En ce début de nouvelle année, nous avons pu constater beaucoup d'animation au sein de la société Trolltech. Toutefois, c'est sans aucun doute le plus importante actualité de Trolltech qui a été diffusée aujourd'hui :

Le géant de la téléphonie mobile Nokia a racheté la société Trolltech pour la somme de 108 millions d'euros !

Cela peut choquer, toutefois les motivations de Nokia semblent bonnes et honnêtes.

Evidemment, la question que la plupart des gens se posent est sûrement Pourquoi Nokia a choisi Trolltech ?

On trouve des éléments de réponses sur le site de Trolltech.

- Nokia souhaite accélérer l'adoption de Qt (ou Qtopia) en tant que framework de développement d'applications pour appareils mobiles, ordinateurs ou toute autre plateforme qui pourrait supporter Qt. La puissance financière de Nokia pourrait en effet apporter un plus à la société Trolltech.
- Etant donnée l'emprise de Nokia sur le monde du mobile ou plus généralement de la communication, co-développer des outils avec Trolltech pourrait mener à d'excellents résultats. On peut en effet imaginer des téléphones Nokia offrant une quantité de possibilités de communication impressionnante, en rajoutant à cela une forte intéractivité avec Internet qui pourrait alors être digne d'un ordinateur. Ce ne sont pour l'instant que des suppositions.

Toutefois, un élément très important ressort de cette affaire. Nokia ne compte pas affecter la relation entre Qt et la communauté opensource. En effet, Nokia a annoncé que Qt continuera à être distribuée avec le système de double-licence actuel. De plus, Nokia aidera Trolltech pour le développement des futures versions de Qt, et non pas seulement pour les produits pour appareils mobiles proposés par Trolltech.

C'est donc une nouvelle très importante qui nous réserve, espérons-le, d'excellentes surprises, que ce soit pour le développement pour appareils mobiles comme pour le développement pour stations PC.

Retrouvez ce billet sur le blog d'Alp Mestan : Lien33

# Les derniers tutoriels et articles

## Interface C ou C++ et Python avec SWIG

Pour programmer une interface entre Python et le C ou le C++ sans utiliser directement l'interface C de Python, on utilise des outils externes tels que SWIG.
Ce texte est issu du livre Python - les fondamentaux du langage - la programmation pour les scientifiques aux éditions ENI. (Lien34)

### 1. Introduction

L'interface C est très riche et par conséquent aussi lourde à gérer. Heureusement, des outils automatiques permettent d'encapsuler des bibliothèques C ou C++ avec peu de code. SWIG est spécialisé dans ce domaine mais sa présentation pourrait faire l'objet d'un livre complet, seule une introduction à son utilisation sera effectuée ici.

SWIG (Lien35) est un utilitaire permettant de créer à partir d'un fichier de configuration **.i** des modules d'encapsulation pour plusieurs langages, dont Python. L'exécutable SWIG crée à partir d'un fichier **module.i** et avec l'option **-python** un fichier **module.py** et un fichier **module_wrap.c** (il est possible d'encapsuler du code C++ avec l'option supplémentaire **-c++** auquel cas le fichier possède une extension **.cpp**). Le fichier **.c** peut alors être compilé en un module appelé **_module.(so**, **dll** ou **pyd** selon la plateforme et la version).

**distutils** est capable de générer un module à partir des fichiers **.i** et des autres fichiers **.c** ou **.cpp** nécessaires. Il est important d'ajouter l'argument **swig_opt=['-c++']** dans la création de l'extension si un code C++ doit être généré.

Contrairement à plusieurs indications sur Internet, SWIG est capable de gérer le code C++ et les cas particuliers qu'il ne supportait pas ont été globalement résolus. En effet, ces critiques portent sur des anciennes versions (la 1.1) tandis que les versions 1.3 actuellement utilisées sont tout à fait adaptées à l'encapsulation de code C ou C++.

Les directives SWIG commencent par **%** et un code SWIG n'est donc pas compilable par un compilateur C ou C++. Si un code SWIG doit être inclus dans un fichier source sans être lu par le compilateur ou si un code C/C++ ne doit pas être lu par SWIG, il est possible de tester la macro **SWIG**. Si elle est définie, SWIG est en train d'analyser le fichier, sinon il s'agit vraisemblablement d'un compilateur.

Lors de la génération du code, toute fonction retournant un résultat complexe par valeur ou prenant des arguments complexes par valeur sera transformée en une fonction retournant un pointeur et prenant en argument des pointeurs. Par complexe, il est entendu tous les types de structures ou de classes non déclarés au moment de l'exécution de SWIG (en effet, celui-ci est plus permissif qu'un compilateur C ou C++).

### 2. Utilisation simple

L'architecture globale d'un fichier **.i** est la suivante :

```
%module mon_module
%{
/* Déclarations supplémentaires */
%}

/*Liste de fonctions, variables ou constantes à inclure
dans le module */
void ma_fonction(void);
```

Tout ce qui se trouve dans le bloc **%{%}** sera intégralement copié dans le fichier source généré, donc toute inclusion d'entête est à déclarer à cet endroit. De même, il est important d'ajouter le ou les fichier(s) d'entête où sont stockés les prototypes des fonctions utilisées. La déclaration **%module** déclare le nom du module à créer, il s'agit couramment du nom du fichier sans l'extension (ceci est une convention). Enfin, une liste de fonctions, de variables ou de constantes qui seront exposées du module doit être écrite. Cette liste d'exposition peut inclure un entête où une sous-partie est déclarée avec la directive **%include** (semblable à la directive du préprocesseur **#include**).

Pour chaque module externe compilé, un module Python est créé. Ce module Python encapsule le module externe, ce dernier possède alors un autre nom, un préfixe **'_'** devant être ajouté au nom du module Python lors de sa compilation. C'est le module Python qui sera utilisé directement, comme cela est préconisé dans le tutoriel sur l'API C.

Quelques directives peuvent spécifier des comportements particuliers :

- **%immutable;** et **%mutable;** interdisent ou autorisent l'écriture sur les attributs suivant la directive
- **%rename(nouveau_nom) ancien_nom;** modifie toutes les références à **ancien_nom** et les remplace par **nouveau_nom** (ce qui est pratique dans le cas où **ancien_nom** est un mot-clé réservé de Python)
- **%ignore nom;** permet d'ignorer toutes les déclarations portant sur **nom**

Lorsqu'une fonction déclare pouvoir prendre en paramètre une autre fonction et que ces fonctions sont exposées dans le module, il n'est pas possible de les utiliser dans Python. Par exemple :

```
void fonction((int (*argument)(int, int)));

int add(int, int);
```

La fonction **add** accessible depuis Python est une version encapsulée de la version C qui n'est donc plus disponible pour être donnée comme argument à **fonction()**. Pour résoudre ce problème, on utilise la directive **%callback("%s_callback")** et

**%nocallback** pour indiquer à SWIG de créer deux objets : le premier est la fonction **add()** classique, la seconde s'appellera **add_callback()** et pourra être donnée comme argument à fonction. Elle n'est en revanche plus appelable depuis Python.

A la place de **"%s_callback"**, il est possible d'utiliser **"%(title)s"** qui utilisera le nom de la fonction avec une majuscule, **"%(upper)s"** pour créer un nom en majuscules uniquement ou **"%(lower)s"** pour uniquement des minuscules. Cette chaîne est semblable à la chaîne de format de la fonction **printf()**.

Un grand intérêt de SWIG est d'autoriser la création de classes à partir d'une structure C. Si une structure a été déclarée, la directive **%extend** permet de créer un bloc dans lequel pourront être définies des méthodes travaillant sur l'instance courante de la structure (dans le cas contraire, SWIG génère lui-même plusieurs wrappers vers les attributs de la structure) :

```
typedef struct structure{
} Structure;
%extend Structure{
  Structure(/* Arguments */)
  {
    Structure* s;
    s = (Structure*) malloc(sizeof(Structure));
    /* Initialisations */
    return s;
  }

  ~Structure()
  {
    free($self);
  }

  void methode()
  {
    /* Methode de la structure */
  }
}
```

L'instance courante dans ces méthodes est **$self**. Pour le constructeur, celle-ci n'existe pas et l'utilisateur doit allouer l'espace mémoire et le désallouer à la destruction de l'objet. En revanche, il est maintenant possible d'exécuter en Python :

```
a = Structure()
a.methode()
```

Le mot-clé **%extend** peut être utilisé au moment de la déclaration de la structure (dans le bloc **struct**). SWIG est aussi capable de détecter les méthodes potentielles parmi les fonctions libres proposées. Par exemple si **new_Structure()** existe, la déclaration du constructeur peut se limiter à **Structure(/* avec les arguments */);**, si **delete_Structure()** prend un seul paramètre, elle pourra être utilisée pour le destructeur directement (pas d'implémentation du destructeur dans **%extend**) et si une fonction **Structure_methode()** dont le premier argument est un pointeur vers une Structure existe, la méthode **methode()** pourra être déclarée sans implémentation dans **%extend** et cette fonction sera appelée automatiquement.

Si la fonction libre se termine en plus par **_get** ou **_set**, SWIG implémentera un attribut en lecture ou en écriture et non une méthode.

Une structure n'a pas besoin d'être déclarée totalement à SWIG. En effet, seul le compilateur par la suite aura besoin d'une définition complète, SWIG ne génère que des fonctions qui vont travailler ensuite sur la structure dans son ensemble.

Si une fonction a besoin d'être déclarée dans le bloc **%{%}** ainsi que dans la liste d'exposition, la directive **%inline** peut être ajoutée avant le bloc **%{%}**. Dans ce cas, SWIG et le compilateur utiliseront ce code (très utile pour déclarer des fonctions utilitaires qui devront être exposées comme des allocations ou des destructions, mais il n'est pas possible d'ajouter des directives SWIG dans ce code puisqu'il sera compilé en C ou en C++).

Une macro SWIG est déclarée par un bloc **%define ma_macro %enddef** dans le cas d'une constante ou pour une fonction **%define ma_macro(ARG1, ARG2) %enddef**.

Avant de passer à une partie plus avancée, quelques règles sont proposées pour créer un module encapsulant un code existant :

- établir une liste précise des fonctions, variables, structures à exposer
- inclure uniquement le minimum utile de fichiers d'entêtes dans les blocs d'inclusion **%{%}**
- créer un fichier d'interface contenant les inclusions des entêtes et les définitions des fonctions nécessaires
- ne pas hésiter à séparer en plusieurs morceaux le fichier d'interface et à inclure les différents sous-fichiers afin d'avoir un fichier d'interface principal clair

## 3. SWIG et le C++

SWIG supporte presque tout le langage C++, à part quelques surcharges telles que celles de **new** et **delete** ou encore les classes imbriquées, mais le reste est intégralement supporté.

Lorsqu'il s'agira de classes, SWIG va générer une classe dite proxy en Python qui va gérer l'interface avec la classe sous-jacente C++. Des mécanismes internes sont proposés pour qu'une instance de la classe proxy puisse être assignée à un attribut d'une classe cible. Par exemple **Classe1** possède un pointeur sur **Classe2**, en Python, il sera possible d'assigner une instance Python à ce pointeur. En général, le mécanisme interne suffit pour gérer correctement la mémoire. Si cela n'est pas le cas, il faudra gérer à la main les cas particuliers à l'aide des méthodes **disown()** et **acquire()** de la classe proxy. Attention aussi, si un objet Python est créé à partir d'un pointeur C++ sur **Classe2**, si le pointeur est détruit (car l'instance de **Classe1** a été détruite), l'objet sera dans un état indéfini car il référencera un objet détruit.

SWIG crée par défaut une encapsulation du constructeur et du destructeur, même s'ils ne sont pas explicitement générés (et que le C++ génère une version par défaut). Pour désactiver cette création, les directives **%nodefaultctor;** et **%nodefaultdtor;** sont proposées, ainsi que **%clearnodefaultctor;** et **%clearnodefaultdtor;** pour réactiver la génération. En ce qui concerne le constructeur de recopie, l'activation se fait par la directive **%copyctor;** et la désactivation par **%nocopyctor;** (par défaut le constructeur de recopie n'est pas créé).

Si les constructeurs ou le destructeur sont protégés ou privés ou si la classe encapsulée est virtuelle pure, ils ne seront pas encapsulés.

En ce qui concerne les attributs et les méthodes statiques, SWIG génère automatiquement le code adéquat, soit des accesseurs soit un appel simple à la fonction statique.

SWIG gère aussi de manière transparente l'héritage, si une classe hérite d'une autre en C++, la version encapsulée de l'une héritera

aussi de la version encapsulée de l'autre.

Enfin, SWIG est capable de gérer dans la limite du possible la surcharge des fonctions en testant le nombre d'arguments et leur type pour savoir quelle fonction ou quelle méthode appeler par la suite.

Si une fonction prenant un entier est surchargée par une fonction prenant en entrée un autre type d'entier, SWIG ne sera pas capable de choisir laquelle utiliser car en Python il n'y a qu'un seul type d'entier (sans compter les entiers longs). L'une des fonctions sera alors choisie et un message d'avertissement affiché. Une solution pour contrer ce problème est d'ignorer la fonction inutile ou de renommer les fonctions.

Un point majeur du C++ est le concept de template. Pour réussir à encapsuler un template (créer un wrapper), il faut l'instancier. Il est toujours possible de le faire explicitement à l'aide d'une déclaration dans la liste d'exposition du template complet instancié pour un type, mais cela est long et le nom du type doit être changé (Python ne supporte pas **vector<int>** comme type de données). Dans ce cas-là, pour un template présenté dans un bloc **%{%}**, la directive **template(vector_int) vector<int>;** permet d'instancier le type **vector** avec le type **int** et d'appeler ce type **vector_int**.

En ce qui concerne les espaces de nom du C++, ils sont tous supprimés et les classes sont exposées dans le module courant (pour exposer un sous espace de nom, un sous-module peut être envisagé). Il est impératif de vérifier si différentes classes dans différents espaces de nom ont des noms identiques.

Les exceptions sont supportées de manière native grâce aux convertisseurs automatiques (présentés un peu plus tard) si les exceptions sont spécifiées dans le prototype de la fonction. Dans le cas où une fonction ou méthode lève une exception qui n'est pas dans cette liste, une directive **%catches()** précédant une déclaration de la fonction indique qu'une certaine série d'exceptions (celles proposées dans la parenthèse) sont transformées en des exceptions Python semblables, et/ou si **...** est donné dans cette parenthèse, toute exception est attrapée et relancée sous la forme d'une exception générique.

Les pointeurs intelligents sont gérés par SWIG. Si une classe expose l'opérateur **->**, tous les attributs et méthodes de la classe pointée seront accessibles par une instance du pointeur intelligent, sauf si l'attribut ou la fonction existe dans le pointeur.

## 4. Création et utilisation de convertisseurs

Plusieurs convertisseurs par défaut sont proposés par SWIG et aussi par Numpy. Les conteneurs de la STL sont définis chacun dans un fichier **std_conteneur.i**. En important ces fichiers, SWIG saura comment convertir le type C ou C++ en type Python.

En ce qui concerne le fichier **exception.i** qui permet de rediriger les exceptions C++ en exceptions Python, une directive permet de changer le mode de conversion :

```
%exception /*une fonction ou méthode particulière à
protéger*/{
  try
  {
    $action
  }
catch(const std::runtime_error& e)
  {
    SWIG_exception(SWIG_RuntimeError,
const_cast<char*>(e.what());
  }
}
```

Par défaut toutes les fonctions ou méthodes seront protégées si aucune fonction ou méthode n'est indiquée. Dans ce cas-ci, SWIG génère l'exception adéquate, le type d'erreur proposé peut être **SWIG_MemoryError**, **SWIG_IOError**, **SWIG_RuntimeError**, **SWIG_IndexError**, **SWIG_TypeError**, **SWIG_DivisionZeroError**, **SWIG_OverflowError**, **SWIG_SyntaxError**, **SWIG_ValueError** ou **SWIG_SystemError**.

Le principal outil de conversion est la directive **%apply** et la directive **%clear** qui stoppe les effets de la première. La directive a cette syntaxe :

```
%apply resultat{type_a_modifier};
/*code à convertir*/
%clear resultat ;
```

Le résultat est le nom de la règle de conversion utilisée et est le type qui remplacera le type donné entre accolades.

Cette solution doit être utilisée pour indiquer qu'un pointeur vers une **std::string** doit être transformé en un **const string&** ou une autre solution basée sur des références. Pour cela, la directive **%apply const string& {std::string*};** automatise toutes les conversions.

Un atout de taille des conversions est de pouvoir spécifier qu'un argument passé en paramètre est un argument de retour, cet argument ne sera pas donné à l'appel en Python mais retourné par la fonction Python (et s'il existe plusieurs arguments, un tuple sera retourné). De même des arguments en entrée et en sortie sont possibles, ils seront toujours retournés et modifiés sur place si leur type le permet.

```
%apply int* OUTPUT {int* resultat};
void ma_fonction(int a, int b, int* resultat);
```

Cette fonction sera appelée ainsi en Python :

```
resultat = ma_fonction(a, b);
```

Une conversion en entrée/sortie est donnée par **INOUT** et une conversion en entrée seulement est donnée par **INPUT**.

Des contraintes peuvent aussi être appliquées :

- **POSITIVE** pour assurer qu'un nombre est strictement positif
- **NONNEGATIVE** pour assurer qu'il est positif ou nul
- **NEGATIVE** pour un nombre strictement négatif
- **NONPOSITIVE** pour un nombre négatif ou nul
- **NONZERO** pour un nombre différent de zéro
- **NONNULL** pour un pointeur non nul.

Si ces contraintes ne sont pas respectées, une exception Python sera levée.

Certaines de ces contraines sont disponibles depuis le fichier **constraints.i**.

Le réel mécanisme de conversion permettant à l'utilisateur de définir une méthode de conversion d'un type en un autre est les typemaps. La syntaxe est la suivante :

```
%typemap(méthode[, modificateurs]) liste_types code;
```

La méthode indique ce que le typemap pourra faire dont les suivantes :

- **in** effectue une conversion du type Python vers le C ou le C++
- **out** effectue l'opération inverse
- **typecheck** permet de vérifier le type donné en entrée (utilisé pour déterminer quelle fonction surchargée doit être utilisée) en retournant 1 si le type est correct et 0 sinon
- **argout** permet d'ajouter à un résultat d'autres valeurs données en argument, le résultat final étant alors un tuple ou une liste (c'est ce qui est utilisé pour les variables passées en argument qui sont en fait des résultats)
- **check** vérifie les valeurs données en entrée (par exemple autoriser uniquement les valeurs positives d'un entier) après la conversion
- **arginit** initialise une donnée avant la conversion (en général, ce n'est pas utile)
- **default** permet d'appliquer une valeur par défaut à l'argument d'une fonction
- **freearg** est appelée à la fin de l'encapsulation pour libérer de la mémoire allouée pendant un typemap in
- **throw** gère les types d'exceptions

La liste de types **list_types** est une liste séparée par des virgules des motifs qui seront validées pour la conversion. Il peut s'agir d'un type simple, d'un type avec un nom d'argument (auquel cas seuls les arguments avec le même nom et le même type seront valides) ou d'une liste de types simples ou avec un nom entre parenthèses (dans le cas où un objet Python serait transformé en plusieurs objets en C ou en C++). Les types qui sont redéfinis avec un typedef dans des espaces de nom différent valident aussi le motif si la définition originale valide ce motif. Enfin, avec la directive **%apply**, il est possible d'appliquer à un nouveau motif un motif défini par un typemap.

Un typemap est valide pour tout le code qui suit jusqu'à ce qu'il soit redéfini ou effacé par **%clear**. En revanche, si %extend est utilisé pour étendre les possibilités d'une classe, le typemap doit être défini avant la définition de la classe.

Les arguments donnés au typemap peuvent être récupérés lors de la déclaration. Par exemple **$n** est l'argument **n** donné dans le motif (il s'agit donc de la destination de la conversion), **$n_name** est son nom, **$n_type** son type et **$n_ltype** son type simplifié (sans mot clé **const**, les tableaux statiques sont transformés en pointeurs, ...). Si **$*** est utilisé à la place de **$**, le type retourné est le type pointé et **$&** rajoute un pointeur. **$n_basetype** indique quant à lui le type sous-jacent simple (**int**, **float**, ...) sans aucun pointeur ni const. Pour des tableaux passés en paramètre dont la taille est fichée, **$n_dimi** donne la taille selon la dimension **i**.

Les typemaps **in**, **out** et **argout** possèdent des variables supplémentaires. La première est **$symname**, contenant le nom de la fonction en cours d'encapsulation. Ensuite **$input** est défini pour **in** et **argout**, et **$output** pour **out** et **argout**.

Si des valeurs temporaires sont nécessaires pour contenir les données pendant la durée d'appel de la fonction, un typemap **in** peut les spécifier entre parenthèses après **liste_types**, avant le code. Même si plusieurs typemaps identiques sont nécessaires pour convertir les données, les variables temporaires ne se chevaucheront pas, SWIG rajoutera un numéro distinctif à la fin du nom de ces variables. Cette variable temporaire peut être accédée dans un typemap **argout** en postfixant le nom de la variable par **$argnum** qui est le numéro de l'argument modifié (accéder à la variable temporaire doit faire l'objet de vérifications minutieuses pour bien être certain de ce qui se passe).

## 5. SWIG avec Numpy

Numpy propose des typemaps préexistants pour SWIG dans le fichier **doc/numpy.i**. Ce fichier permet de définir :

- des tableaux en entrée
- des tableaux modifiés sur place
- des tableaux en sortie mais passés en arguments

Ces typemaps peuvent fonctionner sur un tableau à plusieurs dimensions (3 maximum), sur un pointeur avec les dimensions données avant ou après le pointeur. Par exemple :

```
%apply (int IN_ARRAY2[ANY][ANY]) {(int
matrix[ANY][ANY])};
void ma_fonction(int matrix[3][3]) ;
```

Le mot-clé **ANY** permet d'indiquer que toutes les tailles sont permises. Ici, on applique le typemap indiquant que l'on travaille sur un tableau à 2 dimensions (**IN_ARRAY2**) et lorsque **ma_fonction()** sera encapsulée, un tableau Numpy pourra être passé en paramètre.

Pour les tableaux modifiés sur place, **INPLACE_ARRAY*** sera utilisé et pour les tableaux en sortie, il s'agira de **ARGOUT_ARRAY***. A noter que les tableaux de dimension 2 et 3 ne peuvent être spécifiés que sous la forme **tableau[][]** ou **tableau[][][]** et non sous la forme de pointeurs et de dimensions associées. C'est une limitation des typemaps proposés et non de SWIG.

Pour fonctionner correctement, le module d'extension de Numpy doit être chargé au préalable (la fonction **import_array()**). Pour cela, la séquence suivante est utilisée :

```
%{
#define SWIG_FILE_WITH_INIT
%}
%include "numpy.i"
%init %{
import_array();
%}
```

Le bloc **%init %{%}** colle son contenu directement dans l'initialisation du module proposé. Un seul appel de ce type doit se produire lors du chargement du module, il ne faut pas le copier dans chaque sous-fichier d'interface.

Le fichier d'interface **numpy.i** est un bon exemple complet de typemap mais aussi de définition de macros et de fonctions.

## 6. Exemple d'encapsulation d'une structure avec exposition de l'interface Numpy

Une classe Python complète sera créée, avec possibilité d'affichage et exposition de l'interface Numpy à l'aide de **__array_struct__**.

Voici la structure dans un fichier numpy_swig.h :

```
#ifndef NUMPY_SWIG
#define NUMPY_SWIG

typedef struct _SignedIntBuf
{
  int* data;
```

```
  int shape[2];
  int strides[2];
} SignedIntBuf;
#endif
```

Le fichier d'interface **numpy_swig.i** est maintenant exposé ici en plusieurs parties :

```
%{
#define SWIG_FILE_WITH_INIT
%}
%include "numpy.i"
%init %{
import_array();
%}

%module (docstring="Ceci est un module Python encapsule
par SWIG") numpy_swig
```

Un module est créé et une documentation associée est proposée par l'argument **docstring**.

```
%{
#include "numpy_swig.h"

void delete_SignedIntBuf(SignedIntBuf* buffer)
{
  free(buffer->data);
  free(buffer);
}

void free_array_interface( void* ptr, void *arr )
{
  PyArrayInterface* inter;
  PyObject* arrpy;

  inter = (PyArrayInterface*)ptr;
  arrpy = (PyObject*)arr;
  Py_DECREF(arrpy);
  free(inter);
}
%}
```

Deux fonctions de destruction sont données telles quel dans le code qui sera généré. La première est le destructeur d'un **SignedIntBuf**, structure qui a été définie dans l'entête précédent, et la seconde est la fonction de désallocation.

```
%inline %{PyObject* get__array_struct__(PyObject* self,
int* shape, int* strides, int*data)
{
  PyArrayInterface* inter;
  PyObject* obj;
  int nd;
  nd = 2;

  inter =
(PyArrayInterface*)malloc(sizeof(PyArrayInterface));
  if (inter==NULL)
    return PyErr_NoMemory();

  inter->two = 2;
  inter->nd = nd;
  inter->typekind = 'i';
  inter->itemsize = sizeof(int);
  inter->flags = NPY_NOTSWAPPED | NPY_ALIGNED |
NPY_WRITEABLE;
  inter->strides = strides;
  inter->shape = shape;
  inter->data = data;
  Py_INCREF(self);
  obj = PyCObject_FromVoidPtrAndDesc((void*)inter,
(void*)self, free_array_interface);
  return obj;
}
%}
```

Voici la fonction principale créant l'interface Numpy à partir d'un objet. En réalité, cette fonction prend en paramètre l'objet encapsulé **SignedIntBuf** mais aussi directement les paramètres nécessaires à la création de l'interface comme les dimensions et les données. Rien n'est extrait manuellement ici, c'est SWIG qui devra faire le travail.

Si l'allocation de la mémoire échoue, une exception est levée, celle-ci fait partie des exceptions présentées dans une partie précédente.

Ce code est reporté intégralement dans le code généré et SWIG connaît la fonction, grâce à la directive **%inline**.

Maintenant vient la déclaration des fonctions et variables exportées avec les extensions :

```
%include numpy_swig.h

%extend SignedIntBuf{
  %feature("autodoc", "Une chaîne de
commentaire")SignedIntBuf;
  SignedIntBuf(int width, int height)
  {
    SignedIntBuf* buffer = (SignedIntBuf*)
malloc(sizeof(SignedIntBuf));
    buffer->shape[0] = height;
    buffer->shape[1] = width;
    buffer->strides[0] = width * sizeof(int);
    buffer->strides[1] = sizeof(int);
    buffer->data = (int*)
malloc(width*height*sizeof(int));
    return buffer;
  }

  ~SignedIntBuf();

   char *__str__()
   {
     static char tmp[1024];
     int i, j;
     int used = 0;
     used += sprintf(tmp, "Tableau :\n");
     for(i=0; i < $self->shape[0]; i++)
     {
       for(j=0; j < $self->shape[1]; j++)
         used += sprintf(tmp + used, "%d\t", $self-
>data[j + i*$self->shape[1]]);
       used += sprintf(tmp + used, "\n");
     }
     return tmp;
  }

  %pythoncode
  {
    def __array_struct__get(self):
      return get__array_struct__(self, self.shape,
self.strides, self.data)

    __array_struct__ = property(__array_struct__get,
doc='Array protocol')
  }
};
```

Après avoir inclus l'entête C, une extension de la structure est proposée. Le constructeur est ici exposé entièrement tandis que le destructeur est simplement prototypé puisqu'il a été défini dans une partie précédente.

**%feature("autodoc", valeur);** permet de créer une chaîne de documentation automatiquement pour valeur valant **"0"** ou **"1"**. Ici, une chaîne explicite est donnée pour une fonction particulière.

Une solution est donnée pour retourner une chaîne de caractères décrivant l'objet. Ici, en C, un tableau statique de caractères est utilisé et retourné, mais une chaîne de caractères C++ aurait été plus simple à manipuler.

Enfin, une directive **%pythoncode** est donnée, elle permet de recopier intégralement le bloc dans le code Python généré. Ici cela est nécessaire car **__array_struct__** doit être une propriété Python. De plus, si une méthode C est directement donnée pour l'accesseur, il est impossible de récupérer le pointeur original Python sur l'objet, ce qui est nécessaire pour indiquer que l'objet est référencé dans l'interface (ou on risque une corruption de la mémoire). C'est pourquoi l'accesseur appelle une fonction libre du module avec les paramètres supplémentaires qui seront automatiquement convertis par SWIG en les types sous-jacents.

**7. Conclusion**

Ainsi s'achève cette introduction à SWIG et son utilisation pour Numpy. L'exemple donné en dernière partie est unique sur Internet mais est extrêment utile pour encapsuler un ancien code scientifique et de le réutiliser facilement avec Numpy sans effectuer de copies à chaque utilisation.

C'est la fin de cette première partie consacrée à l'API C. Pour télécharger le code source des exemples ainsi que d'autres exemples, reportez vous à la page consacrée à mon livre (Lien34).

Retrouvez l'article de Matthieu Brucher en ligne : Lien36

# Les livres Python

## Les Fondamentaux du langage - La Programmation pour les scientifiques

L'objectif de ce livre sur Python est de fournir les fondamentaux de ce langage à des scientifiques habitués à d'autres langages comme le Fortran, le C, Matlab... et aussi à des informaticiens qui travaillent dans le milieu scientifique. Des connaissances élémentaires sur la programmation sont nécessaires au lecteur pour tirer le meilleur parti de ce livre.

L'auteur détaille les bases du langage puis présente, avec des exemples, des outils que les scientifiques n'ont pas l'habitude d'utiliser (sauvegarde des données, parallélisme, XML...). Ensuite, l'auteur décrit le module matriciel Numpy (comment manipuler des tableaux, la syntaxe et les fonctions usuelles), un "catalogue" des outils proposés par Scipy (optimisation numérique, statistiques, traitement du signal...) ainsi que des fonctionnalités graphiques indispensables pour les rapports ou les publications. La dernière partie présente les outils pour communiquer avec le C ou le C++.

**Critique du livre par Alp Mestan**

Voici un livre unique en son genre. Comme vous pouvez le voir sur la table des matières, Matthieu Brucher introduit le langage Python en premier lieu, ce qui s'avère très efficace. En effet, ce langage est assez simple à aborder et l'auteur ne fait que faciliter les choses grâce à de bonnes explications, qui vont droit au but, ainsi que grâce à de nombreux exemples.

Ensuite, il s'agit de consolider ses connaissances en python à travers des exemples plus intéressants, qui abordent le XML, le réseau, la compression, le multithreading, etc, ce qui amène le lecteur à faire des essais, à comprendre. J'ai tenu à décrire ce début de livre car c'est à mon avis une très bonne approche qui prépare très bien pour la suite.

C'est à ce moment-là que les choses sérieuses commencent. L'auteur, à partir de ce moment vise particulièrement les scientifiques (bien qu'il ne ferme pas la compréhension aux autres). Matthieu Brucher dédie une grosse partie du livre à des bibliothèques scientifiques utilisables en Python. En effet, une fois la syntaxe et les quelques subtilités du langages acquises, le lecteur a un niveau suffisant pour s'intéresser à la représentation des matrices, la transformée de Fourier, le traitement de signal et tant d'autres choses, en Python. Il s'agit donc à partir de ce moment là de considérer Python non plus comme un "simple langage de programmation" mais comme un outil scientifique pour le calcul, le traitement, la résolution, par exemple. L'auteur aborde donc un large spectre des possibilités de ces bibliothèques Python jusqu'à la fin du livre, en terminant avec l'intéraction entre Python et C/C++.

Ce qui m'a plu dans ce livre, bien que connaissant déjà la programmation (mais peu le langage Python), est la façon dont est abordé le langage. L'élévation du niveau est progressive, mais entraine une motivation due au fait que la programmation se concrétise : on est ensuite capable de se servir de Python comme de Matlab, ou comme tout autre outil de ce genre. De plus, Python étant un langage de programmation simple mais pourtant offrant tellement de possibilités, on peut écrire des applications réellement puissantes : aller chercher un fichier xml à une adresse donnée, récupérer des informations relatives à un signal, construire puis effectuer des traitements sur le signal, tracer différents graphiques, etc. Le sujet choisi par l'auteur est excellent et la mission est réussie. Il y a toutefois une seule chose qui pourrait déranger : ne rien connaître à la programmation ni à l'algorithmie. En effet, l'auteur ne passe pas des pages et des pages à décrire une notion simple. Les explications sont efficaces et seront comprises par, à mon avis, toute personne concernée par ce livre.

Retrouvez cette critique de livre sur la page livres Python : Lien37

# Les derniers tutoriels et articles

## Distribuer vos applications VB6 avec InnoSetup

**InnoSetup** est un logiciel gratuit de déploiement bien connu des développeurs, pour sa simplicité d'utilisation. Ses possibilités sont extraordinaires, et nous allons en découvrir les bases, afin de créer des programmes d'installation "professionnels".

### 1. Introduction

Créer un programme d'installation constitue l'étape finale du développement d'une application.

Compte tenu de ma spécialité, ce tutoriel est forcément très orienté Visual Basic, et principalement la section II.
Mais **InnoSetup** est un installatteur universel, et peut tout à fait être utiliser pour la distribution d'applications conçues avec d'autres languages de programmation (Delphi, par exemple, pour ne citer que celui-ci).

Bien sûr, pour Visual Basic, il existe l'assistant d'installation est de déploiement, livré de base avec avec VB6-édition professionnelle ou +.
Mais la moindre modification étant tellement fastidieuse, qu'elle en devient vite dissuasive !

**InnoSetup** constitue une bonne alternative à cet inconvénient, offrant de multiples possibilités de personnalisation, de façon très simple, sans se prendre la tête à modifier le Setup.bas de VB6 !

**IsTools** est le complément indispensable de **InnoSetup** . **Nécessaire** pour importer facilement les fichiers d'installation **Setup.lst**, généré par l'assistant VB, et disponible en français, il vous permet de personnaliser la base de votre **Setup** sans écrire une seule ligne de script ou de code !

### 2. Mon premier "Setup"

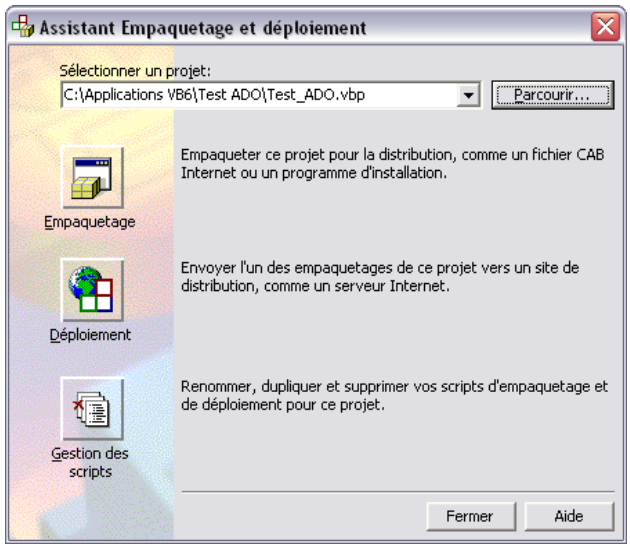
La principale difficulté, pour obtenir un setup complet et opérationnel, est de trouver toutes les dépendances de votre projet VB6.

Si vous n'êtes pas un expert en ce domaine, je vous conseille d'utiliser, dans une première phase, l'assistant de déploiement de VB6, afin d'obtenir un fichier *Setup.lst*, qui contiendra ces dépendances.
Si vous ne disposez pas de l'assistant VB6 (distribué uniquement à partir des version professionnelles de VB6), il existe d'autres solutions, mais nous parlerons plus tard !

#### 2.1. Générer le fichier Setup.lst

Je ne ferai aucun commentaire ici, concernant l'assistant de déploiement VB6.
Nous allons l'utiliser comme simple passerelle, afin d'obtenir la liste des fichiers de dépendences de notre projet.
Pour toute question relative à l'utilisation de ce logiciel, je vous convie sur notre Forum installation et déploiement VB6 (Lien38)

- Lancer ***l'assistant Installateur et déploiement de VB6*** :




Fenêtre d'accueil de l'assistant VB6 Empaquetage et déploiement

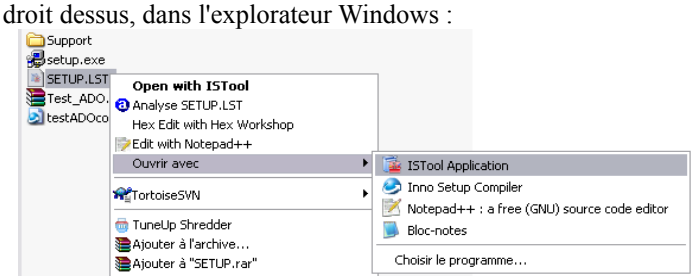
- Sélectionner votre fichier projet VB6 en cliquant sur le bouton "**Parcourir...**" (fichier de type .vbp ou .vbg)
- Choisissez l'option "Empaquetage" et continuez jusqu'à la fin, sans trop vous poser de questions, ce n'est pas le plus important.

A ce stade, (si vous avez conservé les options de base de l'assistant), vous avez généré dans le répertoire *.\MonProjet\Package* un fichier Setup.lst.

Repérez ce fichier, il sera la base de notre installation avec **InnoSetup**.
Certains composants ocx en freeware peuvent avoir des dépendences cachées, non repérées pas l'assistant VB.
Je n'ai pas de solution miracle et ne peux que vous conseiller de bien tester votre Setup avant diffusion (j'en ai fait l'amère expérience)

### 3. Importer un fichier Setup.lst dans InnoSetup/IsTools

Vous avez repéré votre fichier *Setup.LST*, alors faites un click droit dessus, dans l'explorateur Windows :



Puis "**Open with IsTool**" ou "*Ouvrir avec*" >> *IsTools Application*"
- si l'application n'apparaît pas dans la liste, sélectionner "**Choisir le programme...**" et rechercher **IsTools.exe**

Voici ce que nous devons obtenir :

Script d'import IsTools

```
; 'C:\Applications VB6\Test ADO\Package\SETUP.LST'
imported by ISTool version 5.2.1

[Setup]
AppName=MyProgram
AppVerName=MyProgram
DefaultDirName={pf}\MyProgram
DefaultGroupName=MyProgram

[Files]
; [Bootstrap Files]
;
@COMCAT.DLL,$(WinSysPathSysFile),$(DLLSelfRegister),,6/
1/98 1:00:00 AM,22288,4.71.1460.1
Source: COMCAT.DLL; DestDir: {sys}; Flags:
restartreplace uninsneveruninstall sharedfile regserver
; @VB6FR.DLL,$(WinSysPath),,$(Shared),10/2/00 1:00:00
AM,119568,6.0.89.88
Source: VB6FR.DLL; DestDir: {sys}; Flags: promptifolder
sharedfile
;
@STDOLE2.TLB,$(WinSysPathSysFile),$(TLBRegister),,6/3/9
9 1:00:00 AM,17920,2.40.4275.1
Source: STDOLE2.TLB; DestDir: {sys}; Flags:
restartreplace uninsneveruninstall sharedfile
regtypelib
; @ASYCFILT.DLL,$(WinSysPathSysFile),,,3/8/99 1:00:00
AM,147728,2.40.4275.1
Source: ASYCFILT.DLL; DestDir: {sys}; Flags:
restartreplace uninsneveruninstall sharedfile
;
@OLEPRO32.DLL,$(WinSysPathSysFile),$(DLLSelfRegister),,
3/8/99 1:00:00 AM,164112,5.0.4275.1
Source: OLEPRO32.DLL; DestDir: {sys}; Flags:
restartreplace uninsneveruninstall sharedfile regserver
;
@OLEAUT32.DLL,$(WinSysPathSysFile),$(DLLSelfRegister),,
4/12/00 1:00:00 AM,598288,2.40.4275.1
Source: OLEAUT32.DLL; DestDir: {sys}; Flags:
restartreplace uninsneveruninstall sharedfile regserver
;
@msvbvm60.dll,$(WinSysPathSysFile),$(DLLSelfRegister),,
2/23/04 9:42:40 PM,1386496,6.0.97.82
Source: msvbvm60.dll; DestDir: {sys}; Flags:
restartreplace uninsneveruninstall sharedfile regserver

; [Setup1 Files]
; @DBRPRFR.DLL,$(WinSysPath),,$(Shared),7/13/98 1:00:00
AM,33280,6.0.81.63
Source: DBRPRFR.DLL; DestDir: {sys}; Flags:
promptifolder sharedfile
; @stdftfr.dll,$(WinSysPath),,$(Shared),11/21/00
4:46:34 AM,6656,6.0.81.63
Source: stdftfr.dll; DestDir: {sys}; Flags:
promptifolder sharedfile
;
@MSSTDFMT.DLL,$(WinSysPath),$(DLLSelfRegister),$(Shared
),2/23/04 1:00:00 AM,119808,6.1.97.82
Source: MSSTDFMT.DLL; DestDir: {sys}; Flags:
promptifolder regserver sharedfile
;
; >> J'en ai enlevé un peu au milieu !!!
;
;
@msadox.dll,$(WinSysPath),$(DLLSelfRegister),$(Shared),
8/5/04 1:00:00 PM,200704,2.81.1117.0
Source: msadox.dll; DestDir: {sys}; Flags:
promptifolder regserver sharedfile
; @Test_ADO.exe,$(AppPath),,,10/12/07 9:51:35
PM,53248,1.0.0.0
Source: MyProgram.exe; DestDir: {app}; Flags:
promptifolder

[Icons]
Name: {group}\MyProgram; Filename: {app}\MyProgram.exe;
WorkingDir: {app}
```

Les **;** vous montrent les lignes importées du fichier Setup.LST (sous forme de commentaires); ces lignes peuvent être supprimé.

A ce niveau, se situe le principal problème de **IsTools**, et vous l'aurez surement vite remarqué : il n'est pas capable d'inclure le chemin du répertoire source pour les fichiers listés !

La sous-section **[Bootstrap Files]** liste les fichiers du runtime VB6.
Vous pouvez placer ces fichiers dans un répertoire particulier, afin de pouvoir vous en resservir à chaque installation :

Par exemple, ici dans : c:\Redist\VB_ Runtime\

```
[Files]
; [VB Runtime files]
Source: c:\Redist\VB_Runtime\stdole2.tlb; DestDir:
{sys}; Flags: restartreplace uninsneveruninstall
sharedfile regtypelib
Source: c:\Redist\VB_Runtime\msvbvm60.dll; DestDir:
{sys}; Flags: restartreplace uninsneveruninstall
sharedfile regserver
Source: c:\Redist\VB_Runtime\oleaut32.dll; DestDir:
{sys}; Flags: restartreplace uninsneveruninstall
sharedfile regserver
Source: c:\Redist\VB_Runtime\olepro32.dll; DestDir:
{sys}; Flags: restartreplace uninsneveruninstall
sharedfile regserver
Source: c:\Redist\VB_Runtime\asycfilt.dll; DestDir:
{sys}; Flags: restartreplace uninsneveruninstall
sharedfile
Source: c:\Redist\VB_Runtime\comcat.dll; DestDir:
{sys}; Flags: restartreplace uninsneveruninstall
sharedfile regserver
Source: c:\Redist\VB_Runtime\Vb6fr.dll; DestDir: {sys};
Flags: restartreplace uninsneveruninstall promptifolder
sharedfile
```

Pour les autres fichiers, vous devrez définir le répertoire source.

## 3. Personnaliser un peu notre Setup

Il me sera impossible de décrire ici, toutes les possibilités de **InnoSetup**
Nous allons donc nous contenter des plus courantes.

### 3.1. La section [Setup]

Il y aurait beaucoup à dire sur cette section, tant ses possibilités et options sont nombreuses.

Un bref aperçu avec ce que nous avons obtenu lors de la conversion du fichier *Setup.lst*, assorti de commentaires d'explication :

```
[Setup]
; Le nom "commercial" de votre application qui sera
utilisé dans le titre des fenêtres et les dialogues
principaux
AppName=MyProgram
; Le nom et la version de votre programme (visible dans
la page d'accueil)
AppVerName=MyProgram v1.0
;  Le répertoire d'installation par défaut
DefaultDirName={pf}\MyProgram
; Le groupe de programme d'installation par défaut,
dans le menu "Démarrer"
DefaultGroupName=MyProgram
```

Bien d'autres options sont disponibles dans cette section :

Un aperçu des options configurables dans la section [Setup]

```
...
; Définir le nom du programme d'installation
OutputBaseFilename=MyProgram_Setup
; Définir le répertoire d'enregistrement du programme d'installaton compilé
OutputDir=..\Package

; Imposer un compte administrateur pour l'installation (nécessaire pour l'enregistrement de dll et ocx)
PrivilegesRequired=admin

; Fixer les versions minimales de Windows requises pour l'installation
MinVersion=0,5.0.2195

; Définir un répertoire source par défaut dans lequel seront recherché les fichiers de la section [Files],
; si le répertoire source n'est pas précisé
SourceDir=C:\MyProgram

; Afficher un texte d'information avant l'installation
InfoBeforeFile=.\InfosBefore.txt
; Afficher un texte d'information après l'installation
InfoAfterFile=.\InfosAfter.txt
; Afficher une page contenant le texte de licence
LicenseFile=.\Licence.txt

; Définir un Copyright
AppCopyright=Copyright© 2008 thierryaim

; Personnaliser les images des fenêtres de l'installateur
WizardImageFile=C:\Program Files\Inno Setup 5\WizModernImage-IS.bmp
WizardSmallImageFile=C:\Program Files\Inno Setup 5\WizModernSmallImage-IS.bmp
```

Toutes les options de la section **[Setup]** sont aisement configurables dans le menu **"Projet" >> "Options de l'installateur"** de **IsTools**.
Si vous avez la moindre question, rendez-vous sur le forum installation VB (Lien38)

### 3.2. La section [Files]

Elle a été créée automatiquement lors de l'importation du fichier .lst, à l'étape précédente.

Dans cette section, vous pouvez ajouter tous les dossiers ou fichiers devant être joints à l'application :

Principaux paramètres :

- **Source** (obligatoire) : nom (et répertoire) du fichier source
- **DestDir** (obligatoire) : répertoire de destination
- **DestName** (facultatif) : nom du fichier destination, si différent du nom d'origine

```
[Files]
;[Redist Files]
Source: c:\Redist\DCom\dcom98.EXE; DestDir: {app}; Flags: promptifolder deleteafterinstall nocompression; Tasks: DCOM98
; [Application Files]
Source: MyProgram.exe; DestDir: {app}; Flags: promptifolder
Source: MyProgram.ini; DestDir: {app}; Flags: promptifolder
Source: lisezmoi.txt; DestDir: {app}; Flags: promptifolder isreadme
; Ajouter au package un dossier et tout son contenu
Source: images\*.*; DestDir: {app}\images
```

Les principaux Flags de cette section :

- **promptifolder** : Affiche un avertissement si le fichier à installer est plus ancien qu'un fichier identique, déjà présent sur le système cible
- **isreadme** : indique que le fichier est de type lisezmoi. Une boite de dialogue sera automatiquement proposée en fin d'installation
- **sharedfile** : déclare un fichier à installer comme pouvant être partagé par plusieurs applications
- **restartreplace** : provoquera un redémarrage du système si le fichier doit être remplacé
- **regserver** : après installation, indique que le fichier (de type ocx ou dll) doit être enregistré à l'aide de regsvr32 (nécessite PrivilegesRequired=admin)
- **regtypelib** : après installation, indique que le fichier (de type .tlb) doit être enregistré comme nouvelle librairie de type (nécessite PrivilegesRequired=admin)
- **uninsneveruninstall** : précise que le fichier ne doit jamais être désinstallé
- **deleteafterinstall** : fichier devant être effacé après installation
- **allowunsafefiles** : nécessaire, si vous devez inclure à votre package, des fichiers situés dans les répertoires du système

Consulter la liste des fichiers non sûrs à ne pas redistribuer (dépendant des différentes plateformes), dans l'aide en ligne d'**InnoStup**

Pour les autres **Flags**, je vous invite à lire l'aide en ligne de **Inno-Setup**.

### 3.3. La section [Dirs]

permet, lors de l'installation, la création de sous-dossiers du répertoire de l'application

L'exemple ci-dessous crée les dossiers ***data*** et ***bin*** comme sous-dossiers du répertoire de l'application {app} :

```
[Dirs]
Name: {app}\data
Name: {app}\bin
```

### 3.4. La section [Tasks]

Comme son nom l'indique, cette section va nous permettre de définir des tâches à exécuter ou non, suivant les choix de l'utilisateur.

Si cette section est présente dans le script, l'installateur génèrera automatiquement une page de sélection des tâches à accomplir, en fonction des paramètres spécifiés :

```
[Tasks]
Name: InstalRTVB6; Description: Installer le Runtime VB6
Name: DCOM98; Description: D&COM 1.3 (Windows 98); GroupDescription: Compléments:; Flags: checkedonce; MinVersion: 4.1,0; OnlyBelowVersion: 4.9,0
Name: DesktopIcon; Description: Créer une icône sur le &bureau; GroupDescription: Icônes; Flags: checkedonce
Name: QuickLaunchIcon; Description: Créer une icône de &démarrage rapide; GroupDescription: Icônes; Flags: unchecked
```

L'exécution du code sera conditionné par la préselection de la tâche **[Tasks]** :
Reprenons l'exemple de l'installation des fichiers du runtime VB6 :

```
[Files]
; [VB Runtime files]
Source: c:\Redist\VB_Runtime\stdole2.tlb; DestDir: {sys}; Flags: restartreplace uninsneveruninstall sharedfile regtypelib; Tasks: InstalRTVB6
Source: c:\Redist\VB_Runtime\msvbvm60.dll; DestDir: {sys}; Flags: restartreplace uninsneveruninstall sharedfile regserver; Tasks: InstalRTVB6
Source: c:\Redist\VB_Runtime\oleaut32.dll; DestDir: {sys}; Flags: restartreplace uninsneveruninstall sharedfile regserver; Tasks: InstalRTVB6
Source: c:\Redist\VB_Runtime\olepro32.dll; DestDir: {sys}; Flags: restartreplace uninsneveruninstall sharedfile regserver; Tasks: InstalRTVB6
Source: c:\Redist\VB_Runtime\asycfilt.dll; DestDir: {sys}; Flags: restartreplace uninsneveruninstall sharedfile; Tasks: InstalRTVB6
Source: c:\Redist\VB_Runtime\comcat.dll; DestDir: {sys}; Flags: restartreplace uninsneveruninstall sharedfile regserver; Tasks: InstalRTVB6
Source: c:\Redist\VB_Runtime\Vb6fr.dll; DestDir: {sys}; Flags: restartreplace uninsneveruninstall promptifolder sharedfile; Tasks: InstalRTVB6
```

L'installation sera effectuée si et seulement si la tâche ***InstalRTVB6*** a été selectionnée précédement.

### 3.5. La section [Icons]

Cette section permet de définir les icônes à installer.

L'exemple ci-dessous propose la création des icônes :

- principale du programme
- de désinstallation du programme
- pour le fichier "Lisezmoi"
- sur le bureau, conditionné par la tâche *DesktopIcon*, définie précédement dans la section **[Tasks]**
- de lancement rapide, conditionné par la tâche *QuickLaunchIcon*, définie précédement dans la section **[Tasks]**

```
[Icons]
Name: {group}\MyProgram; Filename: {app}\MyProgram.exe; WorkingDir: {app}
Name: {group}\Uninstall MyProgram; Filename: {uninstallexe}
Name: {group}\Lisezmoi; Filename: {app}\Lisezmoi.txt; WorkingDir: {app}
Name: {userdesktop}\MyProgram; Filename: {app}\MyProgram.exe; WorkingDir: {app}; Tasks: DesktopIcon
Name: {userappdata}\Microsoft\Internet Explorer\Quick Launch\MyProgram; Filename: {app}\MyProgram.exe; WorkingDir: {app}; Tasks: QuickLaunchIcon
```

### 3.6. La section [INI]

permet, lors de l'installation, d'écrire dans un fichier .ini associé à l'applicaton :

Paramètres :

- **Filename** : nom du fichier .ini
- **Section** : nom de la section du fichier .ini
- **Key** : nom de la clé du fichier .ini
- **String** : Valeur de la clé

```
[INI]
Filename: {app}\MyProgram.ini; Section: Setup; Key: Language; String: Français; Languages: fr; Flags: createkeyifdoesntexist
Filename: {app}\MyProgram.ini; Section: Setup; Key: Language; String: English; Languages: en; Flags: createkeyifdoesntexist
```

### 3.7. La section [Registry]

permet d'inscrire dans le registre des informations à l'installation :

```
[Registry]
; Création de la clé primaire
Root: HKCU; Subkey: Software\VB and VBA Program Settings\MyProgram; Flags: uninsdeletekey
; Inscription des valeurs de clés secondaires
Root: HKCU; Subkey: Software\VB and VBA Program Settings\MyProgram\Setup; ValueType: string; ValueName: LicenseName; ValueData: {sysuserinfoorg}
Root: HKCU; Subkey: Software\VB and VBA Program Settings\MyProgram\Setup; ValueType: string; ValueName: Language; ValueData: French.lng; Languages: fr
Root: HKCU; Subkey: Software\VB and VBA Program Settings\MyProgram\Setup; ValueType: string; ValueName: Language; ValueData: English.lng; Languages: en
```

### 3.8. La section [Run]

permet de définir les tâches à exécuter lorsque l'intallation est terminée

```
[Run]
; START DCOM
Filename: {app}\dcom98.exe; Parameters: /r:n /q:u; WorkingDir: {tmp}; Flags: skipifdoesntexist; Tasks: DCOM98; MinVersion: 4.1,0
```

## 4. Installation multilingue

Parmi ses nombreuses possibilités, **Inno-Setup** permet, de base, les installation multilingue.

Pour celà, vous devez écrire :

- La section **[Languages]** : permet définir les langues d'installation disponibles pour votre setup.
- La section **[CustomMessages]** : contiendra tous les textes et messages, utilisés dans le setup, dans les différentes langues déclarées

### 4.1. La section [Languages]

La liste des langues pré-formatées est contenue dans le dossier d'installation de Inno-Setup; normalement votre répertoire "C:\Program Files\Inno Setup 5\Languages".
La langue par défaut, vous l'aurez remarqué, est *English*, associée au fichier de langue *default.isl*

Pour une installation en français uniquement, vous n'aurez qu'à définir la langue, en remplacement de la langue par défaut (anglais) :

Pour un installateur qui parle français :

```
[Languages]
Name: fr; MessagesFile: compiler:Languages\French.isl
```

Pour une installation multilingue, vous devez définir un nom (abréviation courte, de préférence) pour chaque langue déclarée.
Nous nous contenterons, dans les exemples ci-après, de deux langues : français et anglais :

Exemple de section [Languages]

```
[Languages]
Name: en; MessagesFile: compiler:Default.isl
Name: fr; MessagesFile: compiler:Languages\French.isl
; Ajouter ci-dessous toutes les langues que vous souhaitez rendre disponibles. Exemples :
Name: gr; MessagesFile: compiler:Languages\German.isl
Name: it; MessagesFile: compiler:Languages\Italian.isl
Name: sp; MessagesFile: compiler:Languages\Spanish.isl
```

Si cette section est présente dans votre programme Setup, et comporte plus d'une langue déclarée, la boite de dialogue de sélection de la langue d'installation sera, par défaut, affichée

automatiquement au lancement du programme :


Boite de dialogue : Selection de la langue d'installation

### 4.2. La section [CustomMessages]

Définir ici tous les textes devant apparaître dans les différentes pages de votre Setup.

Chaque message doit être préfixé du nom (Name) défini dans la section **[Languages]**

Exemple de section [CustomMessages]

```
[CustomMessages]
; Français
fr.TaskIconGroup=Icônes :
fr.TaskDesktopIcon=Créer une icône sur le &bureau
fr.TaskQuickLaunchIcon=Créer une icône de &démarrage rapide

;English
en.TaskIconGroup=Icons :
en.TaskDesktopIcon=Create a desktop icon
en.TaskQuickLaunchIcon=Create a quicklaunch icon
```

Ensuite, dans le script, remplacer les messages texte par leur équivalent avec la syntaxe : {cm:monmessage]

Le texte sera affiché en fonction de la langue choisie au démarrage de l'installation.

Exemple avec la section [Tasks]

```
[Tasks]
Name: DesktopIcon; Description: {cm:TaskDesktopIcon}; GroupDescription: {cm:TaskIconGroup}; Flags: checkedonce
Name: QuickLaunchIcon; Description: {cm:TaskQuickLaunchIcon}; GroupDescription: {cm:TaskIconGroup}; Flags: unchecked
```

Section [Files], conditionnée par la langue d'intallation :

```
[Files]
Source: C:\licence.txt; DestName: licence.txt; DestDir: {app}; Languages: fr
Source: C:\license.txt; DestName: license.txt; DestDir: {app}; Languages: en
Source: lisezmoi.txt; DestDir: {app}; Flags: promptifolder isreadme; Languages: fr
Source: readme.txt; DestDir: {app}; Flags: promptifolder isreadme; Languages: en
```

Si vous êtes amené a créer des pages personnalisées, pour l'utilisation des {cm:..} dans la section **[Code]**, je vous engage à lire la section [Code] de InnoSetup (Lien39)

### 5. Conclusions

A chacun ses préférences, mais moi, je suis un inconditionnel de ce logiciel, simple, facile de prise en main, peu gourmand en ressources, et aux possibilités et options immenses.

Pour les adeptes de la langue de Shakespeare, je vous engage à lire l'aide de **InnoSetup**, pour toutes informations complémentaires, principalement sur les **Flags**, qu'il me serait difficile de décrire ici, tant les possibilités sont nombreuses.

Retrouvez l'article de ThierryAIM en ligne : Lien40

# La section [Code] de InnoSetup

**InnoSetup** est un logiciel gratuit de déploiement bien connu des développeurs, pour sa simplicité d'utilisation. Toutefois, il offre des possibilités extraordinaires, à qui veut bien se pencher sur ses rouages, et particulièrement la section [Code], qui permet de faire *presque* tout ce que l'on veut ! Ce tuto n'est pas exhaustif et n'a pas la prétention de remplacer l'aide d'**InnoSetup**, très bien faite, mais malheureusement, uniquement en anglais

### 1. Introduction

Tout d'abord, si vous n'avez pas ou ne connaissez pas **InnoSetup**, pour pouvez le télécharger gratuitement sur le site de l'éditeur (Lien41 ou par l'intermédiaire de la pages Outils (Lien42) du forum VB6 (Lien43)

Le principal handicap de **InnoSetup**, est, pour la communauté francophone, d'être publié et documenté exclusivement en anglais.

Certaines explications, publiés ci-après, sont extraites soit :
- des codes exemples, fournis avec l'installation de InnoSetup
- de la documentation du logiciel, traduite en français par mes soins

Il n'est pas question de détailler ici toutes les fonctions de la **VCL** de **InnoSetup**, mais d'essayer de comprendre comment utiliser la section **[Code]**. Les fonctions et procédures de la VCL seront à rechercher dans l'aide, suivant ce que vous souhaitez exécuter.

J'adresse ici un tout grand **Merci** aux concepteurs de ce logiciel, dont je suis l'un des plus grand fan.

J'espère que ce petit tutoriel sans prétention vous aidera à utiliser au mieux la puissance de ce logiciel.

Pour toutes informations complémentaires, je vous engage à consulter l'aide d'**InnoSetup**, très bien faite, pour qui maîtrise la langue de Shakespeare. Tout ce que je sais de ce logiciel, je l'ai appris de là, des exemples fournis lors de l'installation ... *(et de quelques prises de tête !)*

### 1.1. Pré requis indispensables :

- une bonne pratique du logiciel **InnoSetup** (connaissance des autres sections standard de script)
- un minimum de connaissance du langage Pascal ou Delphi (principe et syntaxe)

### 2. La section [Code]

La section **[Code]** d'un script **InnoSetup** utilise le langage **Delphi**, le logiciel étant lui-même écrit avec ce langage.
Je ne m'attarderai pas sur la syntaxe de ce langage, ni sur la partie script standard de InnoSetup. (cf. Pré requis)

Nous ne traiterons pas non plus dans ce tutoriel, de l'extension **InnoSetup Préprocesseur**, qui fera (plus tard ...) l'objet d'un prochain cours.

### 2.1. Que peut-on faire avec la section [Code] de InnoSetup ?

A peu près tout ce que l'on veut lors de l'installation d'un progiciel !

- Modifier les pages standard
- Ajouter des pages pré formatées
- Créer de toutes pièces des pages personnalisées
- Dialoguer avec le script
- Créer ses fonctions personnalisées
- ....

Les chapitres suivants vont essayer de vous aider à tirer partie du meilleur de ces fonctionnalités.

## 3. Les fonctions d'évènements

La section **[Code]** est dite évènementielle, c'est à dire qu'elle réagit aux interruptions d'évènements, comme tout langage de ce type (VB, Delphi, ...)

Les évènements sont de 2 types :

- Les évènements standards : générés automatiquement par le script
- Les évènements personnalisés : ceux que vous allez créer

### 3.1. Les évènements standards

Certains évènements sont déclenchés lors du déroulement du script.
Utilisez les fonctions d'évènements afin de modifier le comportement de votre installateur.

Je ne vais pas récrire entièrement l'aide de **InnoSetup**, mais la liste ci-dessous vous permettra d'avoir une idée de quoi et où chercher :

Les évènements standards d'installation de InnoSetup :

- ***function InitializeSetup(): Boolean;*** : appelé lors de l'initialisation du setup (c'est le 1er évènement qui se déclenche)
- ***procedure InitializeWizard();*** : procédure d'initialisation générale
- ***procedure DeinitializeSetup();*** : appelé en fin d'installation (c'est le dernier évènement qui se déclenche)
- ***procedure CurStepChanged(CurStep: TSetupStep);*** : utilisez cet évènement afin de réaliser vos propres pré-install et post-install
- ***function NextButtonClick(CurPageID: Integer): Boolean;*** : se déclenche lors de l'appui sur le bouton **Suivant**
- ***function BackButtonClick(CurPageID: Integer): Boolean;*** : se déclenche lors de l'appui sur le bouton **Précédent**
- ***procedure CancelButtonClick(CurPageID: Integer; var Cancel, Confirm: Boolean);*** : se déclenche lors de l'appui sur le bouton **Annuler**
- ***function ShouldSkipPage(PageID: Integer): Boolean;*** :
- ***procedure CurPageChanged(CurPageID: Integer);*** : se déclenche lors de l'affichage d'une nouvelle page
- ***function CheckPassword(Password: String): Boolean;*** : si présente dans le code, provoque l'affichage d'un boite de saisie d'un mot de passe
- ***function NeedRestart(): Boolean;*** : retourne Vrai afin d'informer l'utilisateur que la désinstallation nécessite un redémarrage
- ***function UpdateReadyMemo(Space, NewLine, MemoUserInfoInfo, MemoDirInfo, MemoTypeInfo, MemoComponentsInfo, MemoGroupInfo, MemoTasksInfo: String): String;***
- ***procedure RegisterPreviousData(PreviousDataKey: Integer);*** :
- ***function CheckSerial(Serial: String): Boolean;*** : si présente dans le code, provoque l'affichage d'un champ de saisie d'un numéro de série sur la page UserInfo
- ***function GetCustomSetupExitCode: Integer;*** :
-

Les évènements standards de désinstallation de InnoSetup :

- ***function InitializeUninstall(): Boolean;*** : retourne Faux pour abandonner la désinstallation, sinon Vrai
- ***procedure DeinitializeUninstall();*** : se déclenche en fin de désinstallation
- ***procedure CurUninstallStepChanged(CurUninstallStep: TUninstallStep);*** : identique à CurStepChanged, mais lors de la désinstallation
- ***function UninstallNeedRestart(): Boolean;*** : retourne Vrai afin d'informer l'utilisateur que la désinstallation nécessite un redémarrage

Le projet standard '**CodeExample1.iss**' montre comment utiliser la plupart de ces évènements

### 3.2. Les évènements personnalisés

Il est bien sûr possible de créer ses propres procédures ou fonctions d'évènement afin de les affecter aux contrôles des pages personnalisées que nous allons construire par la suite.

Exemple de procédure d'évènement pour le click d'un bouton :

```
[Code]
procedure ButtonOnClick(Sender: TObject);
begin
  MsgBox('Vous avez cliquer sur le bouton !',
mbInformation, mb_Ok);
end;
```

La liaison du contrôle et de son évènement se fera par l'intermédiaire d'une propriété d'évènement et de l'opérateur @

Lier la procédure d'évènement au contrôle :

```
Button.OnClick := @ButtonOnClick;
```

Vous trouverez ci-après des exemples complets d'utilisation des fonctions d'évènements personnalisés.

A l'attention des "Delphistes" (suggéré par mon compère sjrd) :
Les procédures et fonctions définies dans la section **[Code]** sont en réalité des méthodes (et non des routines) : **procedure of object** ou **function of object**.
C'est pour cela que vous pouvez les assigner aux événements standard des composants.

Toutefois, le Self n'est pas accessible : il est utilisé en interne pour contenir des données spéciales, dont vous n'avez pas envie de connaître la nature, croyez-moi... ^^

Retrouvez la suite de l'article de ThierryAIM en ligne : Lien39

# Introduction à la programmation en Bash

## 0. Présentation

Interpréteur de commandes par défaut des systèmes GNU/Linux, bash est devenu pour les administrateurs système, un outil incontournable. Ce document présente les principales constructions syntaxiques de bash utilisées dans l'écriture des programmes shell (scripts shell). L'objectif premier a été de laisser de côté les redondances syntaxiques de ce langage de programmation, la subtilité des mécanismes de l'interpréteur, afin d'insister sur quelques concepts synthétiques tels que la substitution, la redirection ou le filtrage.

Cet écrit étant destiné principalement aux étudiants de premier et deuxième cycle en informatique, j'ai choisi de le rédiger en adoptant un style 2Ex (1 Explication, 1 Exemple) qui devrait permettre au lecteur de mieux s'approprier chaque notion présentée. Ce dernier pourra ensuite aborder des publications plus extensives ou plus spécialisées.
Cette publication étant loin d'être parfaite1, j'encourage le lecteur à me faire parvenir ses remarques ou suggestions, ainsi qu'à me signaler toute inexactitude (sanchis@iut-rodez.fr).

## 1. Introduction à bash

### 1.1. Les shells

Sous Unix, on appelle shell l'interpréteur de commandes qui fait office d'interface entre l'utilisateur et le système d'exploitation. Les shells sont des interpréteurs : cela signifie que chaque commande saisie par l'utilisateur (ou lue à partir d'un fichier) est syntaxiquement vérifiée puis exécutée.

Il existe de nombreux shells qui se classent en deux grandes familles :

- la famille C shell (ex : csh, tcsh)
- la famille Bourne shell (ex : sh, bash, ksh).

zsh est un shell qui contient les caractéristiques des deux familles précédentes. Néanmoins, le choix d'utiliser un shell plutôt qu'un autre est essentiellement une affaire de préférence personnelle ou de circonstance. En connaître un, permet d'accéder aisément aux autres. Lorsque l'on utilise le système GNU/Linux (un des nombreux systèmes de la galaxie Unix), le shell par défaut est bash (Bourne Again SHell). Ce dernier a été conçu en 1988 par Brian Fox dans le cadre du projet GNU. Aujourd'hui, les développements de bash sont menés par Chet Ramey.

Un shell possède un double aspect :

- un aspect environnement de travail
- un aspect langage de programmation.

#### 1.1.1. Un environnement de travail

En premier lieu, un shell doit fournir un environnement de travail agréable et puissant. Par exemple, bash permet (entre autres) :

- le rappel des commandes précédentes (gestion de l'historique) ; cela évite de taper plusieurs fois la même commande
- la modification en ligne du texte de la commande courante (ajout, retrait, remplacement de caractères) en utilisant les commandes d'édition de l'éditeur de texte vi ou emacs
- la gestion des travaux lancés en arrière-plan (appelés jobs) ; ceux-ci peuvent être démarrés, stoppés ou repris suivant les besoins
- l'initialisation adéquate de variables de configuration (chaîne d'appel de l'interpréteur, chemins de recherche par défaut) ou la création de raccourcis de commandes (commande alias).

Illustrons cet ajustement de configuration par un exemple. Le shell permet d'exécuter une commande en mode interactif ou bien par l'intermédiaire de fichiers de commandes (scripts). En mode interactif, bash affiche à l'écran une chaîne d'appel (appelée également prompt ou invite), qui se termine par défaut par le caractère # suivi d'un caractère espace pour l'administrateur système (utilisateur root) et par le caractère $ suivi d'un caractère espace pour les autres utilisateurs. Cette chaîne d'appel peut être relativement longue.

```
sanchis@jade:/bin$
```

Celle-ci est constituée du nom de connexion de l'utilisateur (sanchis), du nom de la machine sur laquelle l'utilisateur travaille (jade) et du chemin absolu du répertoire courant de l'utilisateur (/bin). Elle indique que le shell attend que l'utilisateur saisisse une commande et la valide en appuyant sur la touche entrée. Bash exécute alors la commande puis réaffiche la chaîne d'appel. Si l'on souhaite raccourcir cette chaîne d'appel, il suffit de modifier la valeur de la variable prédéfinie du shell PS1 (Prompt Shell 1).

```
sanchis@jade:/bin$ PS1='$ '
$ pwd
  /home/sanchis       => pwd affiche le chemin absolu
du répertoire courant
$
```

La nouvelle chaîne d'appel est constituée par le caractère $ suivi d'un caractère espace.

#### 1.1.2. Un langage de programmation

Les shells ne sont pas seulement des interpréteurs de commandes mais également de véritables langages de programmation. Un shell comme bash intègre :

- les notions de variable, d'opérateur arithmétique, de structure de contrôle, de fonction, présentes dans tout langage de programmation classique, mais aussi
- des opérateurs spécifiques (ex : |, ;)

```
$ a=5                    => affectation de la valeur 5 à
la variable a
$
$ echo $((a +3 ))        => affiche la valeur de
l'expression a+3
8
$
```

L'opérateur |, appelé tube, est un opérateur caractéristique des shells et connecte la sortie d'une commande à l'entrée de la commande suivante.

```
$ ruptime
iutbis          up
jade            up
mobile1         up
mobile2         up
quartz          up
sigma           up
$
$ ruptime | wc -l
6
$
```

La commande unix ruptime affiche les noms et autres informations relatives aux machines visibles sur le réseau. La commande unix wc munie de l'option l affiche le nombre de lignes qu'elle a été en mesure de lire. (La commande ruptime est contenue dans le paquetage rwho. Ce paquetage n'est pas systématiquement installé par les distributions Linux.)
En connectant avec un tube la sortie de ruptime à l'entrée de la commande wc –l, on obtient le nombre de machines visibles du réseau.

Même si au fil du temps de nouveaux types de données comme les entiers ou les tableaux ont été introduits dans certains shells, ces derniers manipulent essentiellement des chaînes de caractères : ce sont des langages de programmation orientés chaînes de caractères. C'est ce qui rend les shells à la fois si puissants et si délicats à utiliser.

L'objet de ce document est de présenter de manière progressive les caractéristiques de bash comme langage de programmation.

### 1.1.3. Atouts et inconvénients des shells

L'étude d'un shell tel que bash en tant que langage de programmation possède plusieurs avantages :

- c'est un langage interprété : les erreurs peuvent être facilement localisées et traitées ; d'autre part, des modifications de fonctionnalités sont facilement apportées à l'application sans qu'il soit nécessaire de recompiler et faire l'édition de liens de l'ensemble
- le shell manipule essentiellement des chaînes de caractères: on ne peut donc construire des structures de données complexes à l'aide de pointeurs, ces derniers n'existant pas en shell. Ceci a pour avantage d'éviter des erreurs de typage et de pointeurs mal gérés. Le développeur raisonne de manière uniforme en termes de chaînes de caractères
- le langage est adapté au prototypage rapide d'applications : les tubes, les substitutions de commandes et de variables favorisent la construction d'une application par assemblage de commandes préexistantes dans l'environnement Unix
- c'est un langage « glu » : il permet de connecter des composants écrits dans des langages différents. Ces derniers doivent uniquement respecter quelques règles particulièrement simples. Le composant doit être capable : de lire sur l'entrée standard, d'accepter des arguments et options éventuels, d'écrire ses résultats sur la sortie standard, d'écrire les messages d'erreur sur la sortie standard dédiée aux messages d'erreur.

Cependant, bash et les autres shells en général ne possèdent pas que des avantages :

- issus d'Unix, système d'exploitation écrit à l'origine par des développeurs pour des développeurs, les shells utilisent une syntaxe « ésotérique » d'accès difficile pour le débutant
- l'oubli ou l'ajout d'un caractère espace provoque facilement une erreur de syntaxe
- bash possède plusieurs syntaxes pour implanter la même fonctionnalité, comme la substitution de commande ou l'écriture d'une chaîne à l'écran. Cela est principalement dû à la volonté de fournir une compatibilité ascendante avec le Bourne shell, shell historique des systèmes Unix
- certains caractères spéciaux, comme les parenthèses, ont des significations différentes suivant le contexte ; en effet, les parenthèses peuvent introduire une liste de commandes, une définition de fonction ou bien imposer un ordre d'évaluation d'une expression arithmétique. Toutefois, afin de rendre l'étude de bash plus aisée, nous n'aborderons pas sa syntaxe de manière exhaustive ; en particulier, lorsqu'il existera plusieurs syntaxes pour mettre en oeuvre la même fonctionnalité, une seule d'entre elles sera présentée.

### 1.1.4. Shell utilisé

La manière la plus simple de connaître le shell que l'on utilise est d'exécuter la commande unix ps qui liste les processus de l'utilisateur :

```
$ ps
PID   TTY   TIME    CMD
6908 pts/4 00:00:00 bash       => l'interpréteur
utilisé est bash
6918 pts/4 00:00:00 ps
$
```

Comme il existe plusieurs versions de bash présentant des caractéristiques différentes, il est important de connaître la version utilisée. Pour cela, on utilise l'option –-version de bash.

```
$ bash --version
GNU bash, version 3.1.14(1)-release (i486-pc-linux-gnu)
Copyright (C) 2005 Free Software Foundation, Inc.
$
```

La dernière version majeure de bash est la version 3. C'est celle qui sera étudiée.

## 1.2. Syntaxe d'une commande

La syntaxe générale d'une commande (unix ou de bash) est la suivante :

```
[ chemin/]nom_cmd [ option ... ] [ argument ... ]
```

C'est une suite de mots séparés par un ou plusieurs blancs. On appelle blanc un caractère tab (tabulation horizontale) ou un caractère espace.
Un mot est une suite de caractères non blancs. Cependant, plusieurs caractères ont une signification spéciale pour le shell et provoquent la fin d'un mot : ils sont appelés méta-caractères (ex :

|, <).
Bash utilise également des opérateurs de contrôle (ex : (, ||) et des mots réservés pour son propre fonctionnement :

```
!     case  do       done elif else   esac
fi    fo    function if   in   select then
time until while    {    }    [[     ]]
```

Bash distingue les caractères majuscules des caractères minuscules.

Le nom de la commande est le plus souvent le premier mot.

Une option est généralement introduite par le caractère tiret (ex : -a) ou dans la syntaxe GNU par deux caractères tiret consécutifs (ex : --version). Elle précise un fonctionnement particulier de la commande.
La syntaxe [ elt ... ] signifie que l'élément elt est facultatif (introduit par la syntaxe [elt ]) ou bien présent une ou plusieurs fois (syntaxe elt ...). Cette syntaxe ne fait pas partie du shell ; elle est uniquement descriptive (méta-syntaxe).

Les arguments désignent les objets sur lesquels doit s'exécuter la commande.

```
ls -l RepC RepD : commande ls avec l'option l et les
arguments RepC et RepD
```

Lorsque l'on souhaite connaître la syntaxe ou les fonctionnalités d'une commande cmd (ex : ls ou bash) il suffit d'exécuter la commande man cmd (ex : man bash). L'aide en ligne de la commande cmd devient alors disponible.

### 1.3. Commandes internes et externes

Le shell distingue deux sortes de commandes :
- les commandes internes
- les commandes externes.

#### 1.3.1. Commandes internes

Une commande interne est une commande dont le code est implanté au sein de l'interpréteur de commande. Cela signifie que, lorsqu'on change de shell courant ou de connexion, par exemple en passant de bash au C-shell, on ne dispose plus des mêmes commandes internes. Exemples de commandes internes : cd , echo , for , pwd Sauf dans quelques cas particuliers, l'interpréteur ne crée pas de processus pour exécuter une commande interne. Les commandes internes de bash peuvent se décomposer en deux groupes :
- les commandes simples (ex : cd, echo)
- les commandes composées (ex : for, ((, {).

**Conmandes simples**

Parmi l'ensemble des commandes internes, echo est l'une des plus utilisée :
echo : Cette commande interne affiche ses arguments sur la sortie standard en les séparant par un espace et va à la ligne.

```
$ echo bonjour     tout le      monde
bonjour tout le monde
$
```

Dans cet exemple, echo est invoquée avec quatre arguments : bonjour, tout, le et monde. On remarquera que les espacements entre les arguments ne sont pas conservés lors de l'affichage : un seul caractère espace sépare les mots affichés. En effet, le shell prétraite la commande, éliminant les blancs superflus.
Pour conserver les espacements, il suffit d'entourer la chaîne de caractères par une paire de guillemets :

```
$ echo "bonjour         tout le
monde"
bonjour       tout le
monde
$
```

On dit que les blancs ont été protégés de l'interprétation du shell.
Pour afficher les arguments sans retour à la ligne, on utilise l'option -n de echo.

```
$ echo -n bonjour
bonjour$
```

La chaîne d'appel $ est écrite sur la même ligne que l'argument bonjour.

**Commandes composées**

Les commandes composées de bash sont :

```
case ... esac        if ... fi             for ... done
select ... done
until ... done       while ... done        [[ ... ]]
( ... )
{ ... }              (( ... ))
```

Seuls le premier et le dernier mot de la commande composée sont indiqués. Les commandes composées sont principalement des structures de contrôle.

#### 1.3.2. Commandes externes

Une commande externe est une commande dont le code se trouve dans un fichier ordinaire. Le shell crée un processus pour exécuter une commande externe. Parmi l'ensemble des commandes externes que l'on peut trouver dans un système, nous utiliserons principalement les commandes unix (ex : ls, mkdir, vi, sleep) et les fichiers shell.

La localisation du code d'une commande externe doit être connue du shell pour qu'il puisse exécuter cette commande. A cette fin, bash utilise la valeur de sa variable prédéfinie PATH. Celle-ci contient une liste de chemins séparés par le caractère : (ex : /bin:/usr/bin). Par exemple, lorsque l'utilisateur lance la commande unix cal, le shell est en mesure de l'exécuter et d'afficher le calendrier du mois courant car le code de cal est situé dans le répertoire /usr/bin présent dans PATH.

```
$ cal
   septembre 2006
di lu ma me je ve sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
$
```

pour connaître le statut d'une commande, on utilise la commande interne type.

```
$ type sleep
sleep is /bin/sleep          => sleep est une commande
externe
$ type echo
```

```
echo is a shell builtin
$
```

### 1.4. Modes d'exécution d'une commande

Deux modes d'exécution peuvent être distingués :

- l'exécution séquentielle
- l'exécution en arrière-plan.

#### 1.4.1. Exécution séquentielle

Le mode d'exécution par défaut d'une commande est l'exécution séquentielle : le shell lit la commande entrée par l'utilisateur, l'analyse, la prétraite et si elle est syntaxiquement correcte, l'exécute.
Une fois l'exécution terminée, le shell effectue le même travail avec la commande suivante. L'utilisateur doit donc attendre la fin de l'exécution de la commande précédente pour que la commande suivante puisse être exécutée : on dit que l'exécution est synchrone.

Si on tape la suite de commandes :

```
sleep 3 entrée date entrée
```

où entrée désigne la touche entrée, l'exécution de date débute après que le délai de 3 secondes se soit écoulé.

Pour lancer l'exécution séquentielle de plusieurs commandes sur la même ligne de commande, il suffit de les séparer par un caractère ;

```
$ cd /tmp ; pwd; echo bonjour; cd ; pwd
/tmp            => affichée par l'exécution de pwd
bonjour         => affichée par l'exécution de echo
bonjour
/home/sanchis   => affichée par l'exécution de pwd
$
```

Pour terminer l'exécution d'une commande lancée en mode synchrone, on appuie simultanément sur les touches CTRL et C (notées control-C ou ^C).
En fait, la combinaison de touches appropriée pour arrêter l'exécution d'une commande en mode synchrone est indiquée par la valeur du champ intr lorsque la commande unix stty est lancée :

```
$ stty -a
speed 38400 baud; rows 24; columns 80; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D;
eol = <undef>;
...
$
```

Supposons que la commande cat soit lancée sans argument, son exécution semblera figée à un utilisateur qui débute dans l'apprentissage d'un système Unix. Il pourra utiliser la combinaison de touches mentionnée précédemment pour terminer l'exécution de cette commande.

```
$ cat
^C
$
```

#### 1.4.2. Exécution en arrière-plan

L'exécution en arrière-plan permet à un utilisateur de lancer une commande et de récupérer immédiatement la main pour lancer « en parallèle » la commande suivante (parallélisme logique). On utilise le caractère & pour lancer une commande en arrière-plan.

Dans l'exemple ci-dessous, la commande sleep 5 (suspendre l'exécution pendant 5 secondes) est lancée en arrière-plan. Le système a affecté le numéro d'identification 696 au processus correspondant tandis que bash a affecté un numéro de travail (ou numéro de job) égal à 1 et affiché [1]. L'utilisateur peut, en parallèle, exécuter d'autres commandes (dans cet exemple, il s'agit de la commande ps). Lorsque la commande en arrière-plan se termine, le shell le signale à l'utilisateur après que ce dernier ait appuyé sur la touche entrée.

```
$ sleep 5 &
[1] 696
$ ps
PID TTY                TIME CMD
683 pts/0        00:00:00 bash
696 pts/0        00:00:00 sleep
697 pts/0        00:00:00 ps
$                => l'utilisateur a appuyé sur la
touche entrée mais sleep n'était pas terminée
$                => l'utilisateur a appuyé sur la
touche entrée et sleep était terminée
[1]+ Done                              sleep 5
$
```

Ainsi, outre la gestion des processus spécifique à Unix, bash introduit un niveau supplémentaire de contrôle de processus. En effet, bash permet de stopper, reprendre, mettre en arrière-plan un processus, ce qui nécessite une identification supplémentaire (numéro de job).
L'exécution en arrière-plan est souvent utilisée lorsqu'une commande est gourmande en temps CPU (ex : longue compilation d'un programme).

### 1.5. Commentaires

Un commentaire débute avec le caractère # et se termine avec la fin de la ligne. Un commentaire est ignoré par le shell.

```
$ echo bonjour         # l'ami
bonjour
$ echo coucou       #l'    ami
coucou
$ # echo coucou
$
```

Pour que le caractère # soit reconnu en tant que début de commentaire, il ne doit pas être inséré à l'intérieur d'un mot ou terminer un mot.

```
$ echo il est#10
il est#10 heures
$
$ echo bon# jour
bon# jour
$
```

### 1.6. Fichiers shell

Lorsqu'un traitement nécessite l'exécution de plusieurs commandes, il est préférable de les sauvegarder dans un fichier plutôt que de les retaper au clavier chaque fois que le traitement doit être lancé. Ce type de fichier est appelé fichier de commandes ou fichier shell ou encore script shell.

**Exercice 1 :**

1.) A l'aide d'un éditeur de texte, créer un fichier premier contenant les lignes suivantes :

```
#!/bin/bash
#    premier
echo -n "La date du jour est: "
date
```

La notation #! en première ligne d'un fichier interprété précise au shell courant quel interpréteur doit être utilisé pour exécuter le script shell (dans cet exemple, il s'agit de /bin/bash).
La deuxième ligne est un commentaire.

2.) Vérifier le contenu de ce fichier.

```
Ex :    $ cat premier
        #!/bin/bash
        #    premier
        echo -n "La date du jour est: "
        date
        $
```

3.) Pour lancer l'exécution d'un fichier shell fich, on peut utiliser la commande : bash fich

```
Ex :    $ bash premier
        la date du jour est : dimanche 3 septembre
2006, 15:41:26 (UTC+0200)
        $
```

4.) Il est plus simple de lancer l'exécution d'un programme shell en tapant directement son nom, comme on le ferait pour une commande unix ou une commande interne. Pour que cela soit réalisable, deux conditions doivent être remplies :

- l'utilisateur doit posséder les permissions r (lecture) et x (exécution) sur le fichier shell
- le répertoire dans lequel se trouve le fichier shell doit être présent dans la liste des chemins contenue dans PATH.

```
Ex :    $ ls -l premier
        -rw-r--r-- 1 sanchis sanchis 63 sep
3 15:39 premier
        $
```

Seule la permission r est possédée par l'utilisateur.

Pour ajouter la permission x au propriétaire du fichier fich, on utilise la commande :

```
chmod u+x fich
```

```
Ex :  $ chmod u+x premier
      $
      $ ls -l premier
      -rwxr--r-- 1 sanchis sanchis 63 sep             3
15:39 premier
      $
```

Si on exécute premier en l'état, une erreur d'exécution se produit.

```
Ex :  $ premier                                                  =>

Problème !
      -bash: premier: command not found
      $
```

Cette erreur se produit car bash ne sait pas où se trouve le fichier premier.

```
Ex :                                              => affiche
la valeur de la variable PATH
      $ echo $PATH
      /bin:/usr/bin:/usr/bin/X11
      $
      $ pwd
      /home/sanchis
      $
```

Le répertoire dans lequel se trouve le fichier premier (répertoire /home/sanchis) n'est pas présent dans la liste de PATH. Pour que le shell puisse trouver le fichier premier, il suffit de mentionner le chemin permettant d'accéder à celui-ci.

```
Ex :  $ ./premier
      la date du jour est : dimanche 3 septembre 2006,
15:45:28 (UTC+0200)
      $
```

Pour éviter d'avoir à saisir systématiquement ce chemin, il suffit de modifier la valeur de PATH en y incluant, dans notre cas, le répertoire courant (.).

```
Ex :                                  => ajout du
répertoire courant dans PATH
      $ PATH=$PATH:.
      $
      $ echo $PATH
      /bin:/usr/bin:/usr/bin/X11:.
      $
      $ premier
      la date du jour est : dimanche 3 septembre 2006,
15:47:38 (UTC+0200)
      $
```

**Exercice 2 :**

Ecrire un programme shell repcour qui affiche le nom de connexion de l'utilisateur et le chemin absolu de son répertoire courant de la manière suivante :

```
Ex :  $ repcour
      mon nom de connexion est : sanchis
      mon repertoire courant est : /home/sanchis
      $
```

Retrouvez l'article d'Eric Sanchis en ligne : Lien44

# Les blogs Linux

## eCryptfs : Encrypter vos données

Bonjour à tous,

Hier, je me suis posé la question de l'encryptage des filesystems. Je suis tombé sur eCryptfs.
Disponible depuis le noyau 2.6.19, ce système de fichier semble répondre à mes attentes.

Allons donc le tester !

**Première étape** : le Noyau

Et oui, il nous faut un ou deux petits modules en plus.

Comme je ne suis pas du genre "fouillez partout" ...j'ai simplement lu la doc :

```
KERNEL BUILD OPTIONS

Code maturity level options --->
  [*] Prompt for development and/or incomplete
code/drivers

Security options --->
  <M> Enable access key retention support

Cryptographic options --->
  <M> MD5 digest algorithm
  <M> AES cipher algorithms

File systems --->
  Miscellaneous filesystems --->
  <M> eCrypt filesystem layer support (EXPERIMENTAL)
```

**Deuxième étape** : Installation des outils.

Bon je vous avoue, j'ai installé les outils mais d'un autre coté, je ne les ai pas utilisé. Bon pour la forme quand même :

```
cast -c ecryptfs-utils (Source Mage)
apt-get install ecryptfs-utils (Debian)
```

**Troisième étape** : Le cryptage des fs

Créons un répertoire perso :

```
mkdir -p /perso
```

Montage du filesystem :

```
mount -t ecryptfs /perso /perso
```

C'est la que ca devient interessant !

```
root@katyucha:/# mount -t ecryptfs /perso /perso
Select key type to use for newly created files:
 1) openssl
 2) passphrase
Selection: 2
Passphrase:
Verify Passphrase:
```

On insère donc sa passphrase : Prenez quelque chose de long , pas toto hein !

Select key bytes:
1) 16
2) 32
3) 24
Selection [16]: 2

On choisit la longueur de la clé de cryptage : ici 32

```
Attempting to mount with the following options:
  ecryptfs_key_bytes=32
  ecryptfs_cipher=aes
  ecryptfs_sig=de1b13eb2242cd42
WARNING: Based on the contents of [/root/.ecryptfs/sig-
cache.txt],
it looks like you have never mounted with this key
before. This could mean that you have typed your
passphrase wrong.

Would you like to proceed with the mount (yes/no)? yes
Would you like to append sig [de1b13eb2242cd42] to
[/root/.ecryptfs/sig-cache.txt]
in order to avoid this warning in the future (yes/no)?
yes
Successfully appended new sig to user sig cache file
Mounted eCryptfs
```

Et voila, Fini

Maintenant, copiez un fichier dans /perso. Prenons un mp3.
Ouvrez le par mplayer ou votre lecteur préféré. La musique est bonne
Puis :

```
umount /perso
```

Essayez de lire le mp3 ..... et ben non ! Ca ne marche pas Illisible !

Pour remonter /perso, faites

```
mount -t ecryptfs /perso /perso
```

Vous rentrez votre passphrase et c'est bon

**Une remarque pour finir :**
Ne pensez pas cacher vos fichiers illégaux comme cela. Quand le fs est démonté, on voit le nom des fichiers à l'intérieur, certes illisible mais le nom est là.
C'est justement cela, qui me gène finallement, si je perd mon ordinateur portable, je ne veux pas qu'on trouve mes fichiers openoffice titré : compte_rendu_client_machin, propal_client_truc

eCryptfs est bien mais ne répond pas entièrement à mes attentes....
Dommage parce qu'il est tout de même facile a mettre en place ...

Retrouvez ce billet sur le blog de Katyucha: Lien45

# Les derniers tutoriels et articles

## MacWorld 2008 : L'année commence bien pour Apple

Les annonces faites durant le Keynote de l'édition 2008 de MacWorld, commentées par la rédaction Mac.

### 1. Quelques chiffres pour commencer

#### 1.1. Mac OS X Léopard

5.000.000. C'est le nombre de licences de Mac OS X Léopard qui ont été vendues ces 3 derniers mois.

Ce qui fait que Mac OS X Léopard représente déjà 20% de la base installée des Mac OS X. C'est la version de Mac OS X qui a été la plus rapidement adoptée.

#### 1.2. iPhone

Cela faisait déjà 200 jours que l'iPhone a été lancé. Juste 200 jours.
Et en 200 jours, 4 millions d'iPhones ont été vendus. En 200 jours. 4.000.000 en 200 jours, cela nous fait 20000 iPhones par jour, y compris les dimanches et jours fériés. Un iPhone vendu toutes les 4 secondes.
Ce qui fait qu'Apple se retrouve déjà devant Palm, Nokia, Motorola, sur le marché du Smartphones. Et cela, en 200 jours.

#### 1.3. iTunes

4 Milliards de chansons vendues dans le monde. Un nouveau record de vente a été atteint durant la période de fin d'année. En un seul jour, Appel a vendu 20 millions de chansons. En un jour.

iTunes a également déjà vendu 125 millions d'émissions de télévisions, et 7 millions de films.

Steve Jobs a avoué être déçu par le chiffres de 7 millions de films. Il a avoué que cela n'était pas ce qu'il espérait. Mais les choses devraient changer.

### 2. Time Capsule

Steve Jobs, en parlant de Mac OS X Léopard, est revenu sur Time Machine, un logiciel de backup vraiment formidable. Mais, pour le moment, il fallait obligatoirement brancher un disque dur au MacBook pour que Time Machine fasse son travail. Steve Jobs a alors présenté un nouveau compagnon à Time Machine, Time Capsule.


Time Capsule est en fait un AirPort Extreme, intégrant un disque dur de même qualité que ceux qu'on retouve dans les serveurs.


Deux versions existent : l'une à 500Go, vendue 300$/Euros, l'autre à 1To, vendue 500$/Euros
A notez que Time Capsule n'est pas une solution RAID. Donc, oui on a une solution de backup. Mais tout de même pas très sécurisée. Et puis, un backup via les airs, c'est tout de même plus lent qu'un backup filaire.
On aurait également aimé apprendre qu'Apple avait une solution pour tous ceux qui ont déjà une AirPort Extreme et un disque dur branché dessus. Nous espérons qu'ils ne seront pas oubliés.

### 3. Mise à jour pour l'iPhone

Steve Jobs a tout d'abord indiqué qu'il y aurait bien un SDK pour l'iPhone, qui sortirait fin Février. Mais on n'en sait toujours pas plus.

Par contre, il s'est montré beaucoup plus bavard sur les nouveautés logicielles qui seraient accessibles aux possesseurs d'iPhone via iTunes.

- Google Maps avec un système de localisation
- WebClips
- Personalisation de la page d'accueil
- Envoi de SMS à plusieurs personnes en une fois
- Possibilité de naviguer par chapitre dans les vidéos, de choisir les sous-titres et les langues
- Support des paroles dans les chansons.

Pour ce qui est du système de localisation intégré à l'iPhone, Steve Jobs a bien précisé que l'iPhone n'avait toujours pas de GPS intégré à l'iPhone. Mais que le système de localisation travaillait en fait sur base de la triangularisation. La position de l'iPhone est retrouvée non seulement par rapport aux antennes GSM, dont les positions sont reprises dans une base de données alimentée par Google mais aussi par rapport aux HotSpots WIFI, dont les positions sont relevées par une autre société. Société qui a fini les Etats-Unis et le Canada, et qui s'attaque à l'Europe et l'Asie.

Steve Jobs a annoncé que cette mise à jour de l'iPhone n'était que logicielle, et offerte gratuitement à tout possesseurs actuels d'un

iPhone.

## 4. Mise à jour pour l'iPod Touch

L'iPod Touch a également droit à sa mise à jour logicielle, puisque vous aurez également droit à la nouvelle version de Google Maps, avec localisation, mais qui se fera elle uniquement sur base des Hotspots WIFI, à l'application Mail vous permettant de lire et envoyer vos mails, à une application pour voir l'évolution de vos actions, encore une autre pour accéder à la météo, et la dernière permettant de prendre des notes.

Contrairement à l'iPhone, la mise à jour de l'iPod Touch est payante. Et le prix de cette mise à jour s'élève à 20$/Euros.

On se demande tout de même pourquoi la mise à jour de l'iPhone est gratuite, et pas celle de l'iPod.

## 5. Nouveautés sur l'iTunes Store

Comme dit précédemment, Steve Jobs était déçu par les chiffres de vente des films. Et il lui fallait faire quelque chose pour changer cela. Il a donc annoncé qu'à partir de maintenant, il serait possible de louer des films via iTunes, au lieu de devoir les acheter.

Bien évidemment, pour pouvoir louer des films, il fallait signer des accords avec les distributeurs de films. C'est à ce moment-là qu'il a annoncé avoir pu signer des accords avec des sociétés comme :

- Touchstone Pictures
- Miramax Films
- Metro Goldwyn Mayer
- New Line Cinema
- Lion Gates Films


Mais il ne s'est pas arrêté là, puisqu'il a également annoncé avoir signé des accords avec :

- 2Oth Century Fox
- Warner Bross
- Walt Disney
- Paramount
- Universal
- Sony Pictures

Ce qui fait que tous les grands majors de Hollywoods sont là. Ouvrant les portes à la quasi totalité de tous les films provenant d'amérique.

Mais voilà. C'est bien d'avoir des films disponibles depuis l'Apple Store, mais c'est encore mieux de pouvoir la regarder sur la TV. Comment ?

A noter que les vidéos seront disponibles en HD, mais pas en FULL HD.

## 6. Apple TV Take 2

C'est là que Steve Jobs a annoncé la nouvelle version de l'Apple TV : Apple TV Take 2.

Steve Jobs a reconnu humblement que la première version de l'Apple TV ne répondait pas aux attentes des utilisateurs. Apple TV Take 2 est la réponse à ce que les utilisateurs attendent. Voyez plutôt :

- Autonome : plus besoin d'être raccordé à un Mac ou un PC
- Possibilité de louer directement les films depuis l'Apple TV, en passant par l'iTunes
- Les films seront disponibles en qualité HD et Dolby 5.1
- Possibilité d'écouter et visualiser les PodCasts audio/vidéos disponibles gratuitement sur iTunes
- Possibilité de voir les photos disponibles sur Flickr et .MAC
- Possibilité de voir les plus de 5 millions de vidéos disponibles sur YouTube
- Possibilité d'acheter les émissions TV et la music
- Possibilité de voir ou d'entendre tout ce qui se trouve sur votre PC ou Mac

Bref, l'Apple TV devient l'interface de salon à iTunes, YouTube, Flick, .MAC.

Steve Jobs indique également que tout cela sera disponible via une mise à jour logicielle, offerte gratuitement aux possesseurs de la première version de l'Apple TV.

Le prix de l'Apple TV baisse également puisqu'il passe de 300$/Euros à 230$/Euros pour la version de base.

Mais l'Apple TV ne permet toujours pas de jouer les DIVX. Ce n'est donc qu'une réponse partielle à ce que les utilisateurs attendent.

## 7. MacBook Air

La dernière grande annonce concernait le nouveau portable, MacBook Air, confirmant les rumeurs des derniers jours.


L'image ci-dessus ne reprend pas les caractéristiques du MacBook Air, mais celles du portable le plus fin jusqu'à présent. Mais Steve Jobs a quelque peu critiqué les choix de son concepteurs, disant que la seule chose avec laquelle il était d'accord, c'était le poids de 3 livres, (+- 1,5Kg), mais que lui ne voulait pas de compromis sur la taille de l'écran, le clavier, la puissance. Et qu'il voulait encore plus fin.
C'est ainsi qu'il a présenté le portable comme étant le plus fin au monde. Jugez plutôt : l'épaisseur du portable coté arrière est plus fine que l'épaisseur de l'autre portable coté avant. Le MacBook Air ne fait que 0,4 cm de haut à l'avant, et 1.9 cm de haut à l'arrière.

Mais qu'a-t-il dans le ventre ?

- Poids : +- 1.5Kg (3 livres)
- Le plus fin : 0.4cm à l'avant, 1.9 cm à l'arrière
- Ecran de 13.3 pouces. La même taille que celle du MacBook
- Caméra iSight située au dessus de l'écran
- Clavier de taille normal commle sur le MacBook de dernière génération, mais rétro-éclairé comme sur le MacBook Pro
- 2Go de RAM en standard
- Disque dur de 80Go
- Possibilité d'avoir un SSD de 64Go à la place
- Processeur Core 2 Duo 1.6Ghz (possibilté d'avoir aussi 1.8Ghz)
- Connection Bluetooth, Wi-Fi, USB2, Micro-DVI, Audio Out, MagSafe
- Fermeture Aimantée
- TouchPad gérant de nouvelles possibilités

Point de vue visuel, avec sa coque en aluminium, son clavier noir, et ses rondeurs lui donne une belle allure.

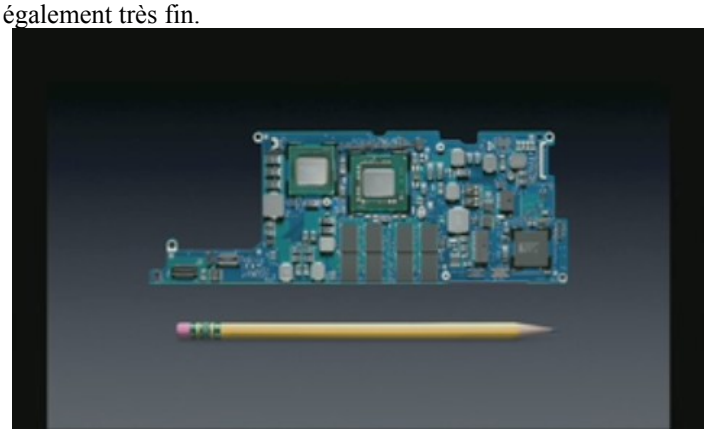
L'écran de 13.3 pouces est rétro-éclairé par des LED, ce qui lui, d'après les premières réactions de ceux qui ont pu l'appocher après la Keynote, permet d'afficher des images aux couleurs sublimes, et de très bonne qualité.

D'après les premières réactions, le clavier a l'air plus dur au toucher que celui du MacBook. Mais cela n'est pas plus génant que cela.

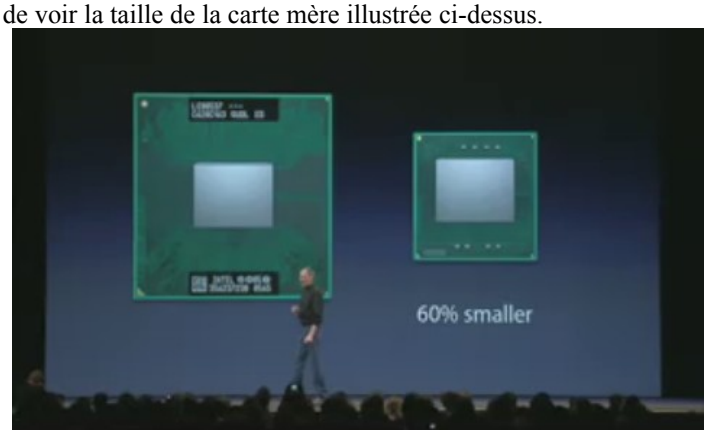
Steve Jobs a expliqué de nouvelles gestuelles possibles qu'on pourra faire avec le MacBook Air, comme par exemple zoomer sur les photos, les faire pivoter, de la même façon qu'avec l'iPhone, mais aussi les faire défiler, cette fois-ci en utilisant 3 doigts. Il sera possible de parametrer tout cela via un nouvelle page de configuration dans Mac OS X Léopard.
On se pose tout de même quelques questions à propos du TouchPad : Les MacBook et MacBookPro gérant déjà le multi-touche (au moins à 2 doigts), pourront-ils avoir une mise à jour logicielle pour tirer profit des nouveaux gestes possibles avec le TouchPad ? Est-ce que les dernières versions de MacBook (Novembre 2007) possèdent déjà le TouchPad de dernière génération comme celui du MacBook Air supportant la reconnaissance des 3 doigts ?
La batterie, de 45 Watts, permet au portable d'avoir une autonomie de 5 heures.

A noter la présence d'un seul port USB2, l'absence de tout port FireWire et même Ethernet. Le port DVI est au format Micro-DVI (celui des autres MacBook/MacBook Pro est au format Mini-DVI)

Il est bien évidemment dépourvu d'un Lecteur Optique. Mais Apple propose un lecteur externe alimenté par le port USB, également très fin.



Techniquement, Apple a vraiment réussi un tour de force. Il suffit de voir la taille de la carte mère illustrée ci-dessus.




Mais pour réussir cette prouesse, Apple n'aurait pu y arriver sans la collaboration d'Intel qui a revu expressement le packaging du processeur, le rendant 60% plus petit.

Tout est tellement intégré et fin, que l'évolution est également quasi nulle. La mémoire de 2Go est soudée. Impossible de passer à 4Go. Rien n'est prévu pour changer le disque dur ou la batterie soi-même. Il est également difficile de réutiliser vos accessoires que vous aviez déjà. Par exemple, il faudra racheter un adaptateur DVI-VGA, vu que c'est un connecteur Micro-DVI et non Mini-DVI comme sur le MacBook ou MacBookPro. Et pourquoi n'y a-t-il pas de Wireless USB sur ce MacBook Air ?

## 8. La réponse d'Apple à GreenPeace

Apple a vraiment intégré l'aspect environnemental dans ses communications, même durant le Keynote. Ainsi, Steve Jobs n'a pas manqué de souligner que l'Ecran LCD ne contenait ni Mercure, ni Arsenic, que le cablage interne ne contenait pas de PVC, que la carte mère ne contenait pas de BFC. Il a même précisé que l'emballage avait encore été réduit par rapport à celui du MacBook.

## 9. Ce que l'on en pense

### 9.1. La force d'Apple

Comment Apple a réussi à convaincre Intel à retravailler le packaging de son processeur ?
Comment Apple a réussi à ce que tous les grands studios de Hollywood soient présents sur iTunes pour ce qui est de la location des films ?
Comment Apple, totalement nouveaux dans le marché du téléphone, occupe déjà 20% du segment du SmartPhone aux états-unis ?
C'est lors de ces annonces-là qu'on se rencontre qu'Apple a

vraiment une force que d'autres sociétés n'ont pas.

### 9.2. Les absents du Keynote

Ceux qui s'attendaient à une mise à jour des MacBookPro doivent être très certainement déçus.

Concernant le SDK pour l'iPhone, bien qu'il soit confirmé, on n'en sait toujours pas plus.

La version 10.5.2 de Léopard n'a toujours pas l'air d'être disponible.

Les mises à jour, de surplus gratuites, de l'iPhone le rendent encore plus intéressant. Finalement, il sera possible de gérer ses mails avec l'iPod Touch. Un manque qui est finalement comblé. On se demande vraiment quel est le public visé par ce nouveau MacBook Air. L'Apple TV 2 devrait rencontrer, même en France, un bien meilleurs succès que le 1er modèle. C'est vraiment l'iTunes qui rentre dans votre salon. Plus besoin d'un PC ou d'un Mac pour l'Apple TV 2.

Retrouvez l'article de la rédaction Mac en ligne : Lien46

# FAQ Mac

## Qu'est-ce que "bonjour" et à quoi sert-il ?

Bonjour était anciennement appelé "Rendez-vous". C'est un protocole proche des DNS(Domain Name Service). En effet son but est de diffuser sur un réseau les services que proposent les ordinateurs qui le composent.

- Apple propose à l'installation un client *bonjour* pour Windows. Il est utilisé lorsque votre Mac partage une imprimante. Les clients windows peuvent ainsi passer par *bonjour* pour rechercher les imprimantes partagées.
- C'est aussi utilisé pour rechercher des sites web diffusant leur adresse par bonjour. Dans Safari, l'onglet bonjour répertorie tous ces sites.
- iTunes, appleTv, les bornes airport, se servent de *bonjour* pour partager leur liste de lecture.
- iChat peut rechercher aussi les autres utilisateurs qui sont sur le réseau.
- ou encore les jeux comme Quinn qui est un tetris-like
- Tous les logiciels peuvent utiliser bonjour pour leur service

Nos amis linuxiens et *BSDiens connaissent bonjour sous le nom de zeroconf.

## Comment trouver facilement l'endroit où je peut configurer une option ?

Vous pouvez rechercher avec Spotlight ou encore dans l'application "Préférences Système". Il y a un champ de recherche. Commencez à taper ce que vous recherchez, et l'application vous proposera ce qui s'en approchera le plus tout en faisant ressortir les sections pouvant héberger les options liées à votre demande.

## Comment reimentionner les champs textes avec Safari 3 ?

Il y a dans le coin inférieur droit des champs textes multi-lignes, trois lignes en diagonal. Faites-les glisser pour agrandir ou pour rétrécir le champ.

## J'ai développé une application en objective-c, mais j'ai une fuite de mémoire, existe t'il des applications qui peuvent m'aider ?

Oui.
Sous Tiger(voir peut être panther), Apple propose "OjectAlloc". A travers cette dernière, vous pourrez lancer votre application et suivre le nombre d'instances des objets créés. Un autre logiciel dans la même trempe: MallocDebug. Vous trouverez cette application avec les developers tools, dans le dossier /Developer/Applications/Performance Tools. Dans ce dossier d'autres applications sont disponibles, allez y faire un tour.

Pour Leopard, ces applications seront obsolètes. En effet ce dernier possédera Xray qui utilise la technologie Dtrace de Sun pour aider à trouver les fuites de mémoire et aider à l'optimisation.

## Comment télécharger la dernière version des outils développeurs ?

Tous les outils développeurs ainsi que la documentation sont disponibles en téléchargement sur le site Developer Connection (Lien47) d'Apple sous réserve de se créer un compte d'accès (gratuit). La suite logicielle est totalement gratuite.

## Existe-t-il un outil plus pratique que l'aide d'Xcode pour parcourir rapidement la documentation des classes Cocoa ?

Oui, le gratuiciel AppKiDo (Lien48) est le logiciel idéal.

## Où est installée la suite développeur car je ne trouve rien dans mon dossier Application ?

Les outils ainsi que la documentation (ADC Reference Library) et des codes d'exemples sont installés dans le répertoire */Developer* à la racine du disque.

## Comment forcer Xcode à s'arrêter sur une exception Objective-C ?

A la racine du répertoire utilisateur, créez un fichier texte nommé ".gdbinit". Editez-le pour y ajouter les lignes suivantes:

```
fb -[NSException raise]
fb objc_exception_throw()
```

Quittez votre session puis relancez XCode. Lorsque le débogueur est lancé, Xcode s'arrêtera de lui même sur les exceptions levées.

## Comment créer une interface graphique avec Xcode ?

Xcode est conçu pour travailler de concert avec Interface Builder qui s'occupe de la création des IHM.

## Je souhaite désinstaller les developper tools, comment dois-je procéder ?

Ouvrez le terminal puis :

```
cd /Developer/Tools
perl uninstall-devtools.pl
```

(perl peut être ignoré)

Notez qu'un sudo est tout de même nécessaire.

## Qt, la librairie développement d'IHM, existe t'elle sur Mac Os ?

Oui, elle est disponible. La licence d'utilisation est identique à celle de linux. Et elle ne passe pas par X11.

## Qu'est-ce que webInspector dans safari ?

C'est un outil qui n'est pas activé par défaut sur safari. Il permet d'accéder à une fenêtre flottante qui détaille les éléments html d'une page web.

## Comment activer webInspector ?

en tapant ceci dans Terminal.app :

```
defaults write com.apple.Safari \
WebKitDeveloperExtras -bool true
```

il sera ensuite accessible par menu contextuel sur une page Web. L'astuce n'a pas été testée sur Safari 2.0.

## Comment éditer les fichiers .plist simplement ?

Vous avez plusieurs solutions:

**1. En ligne de commande :"defaults"**

Pour la lecture :

```
defaults read 'domain' ['key']
```

***domain***: c'est le nom d'un plist de préférence sans son extension et le dossier d'emplacement. exemple : com.apple.Safari.
***key*** : le nom d'une clé que l'on souhaite lire. Pour plus d'information:

```
man defautls
```

dans le terminal.

**2. Utiliser l'application Property List Editor**

C'est un éditeur xml spécialisé dans les *.plist (il ne fait que ça).
Il se trouve dans le dossier : /Developer/Applications/Utilities/.
Il est nécessaire d'avoir les developper tools d'installés.

**3. Utiliser le shareware PlistEdit Pro**

Vous le trouverez à cette adresse: (Lien49)

## Comment compiler un programme en Universal Binaries ?

Par défaut Xcode ne compile pas en UB, mais pour la plateforme sur laquelle le développement est fait. Pour compiler en UB suivez la procédure suivante, dans Xcode :

- faites clic-droit->get info sur votre projet.
- double-clic sur la ligne "Architectures"
- dans la "fenêtre sheet" qui s'ouvre cochez la seconde architecture, puis cliquez sur ok.
- cliquez sur "build" pour re-compiler votre projet au format UB.

## Qu'est-ce que RealBasic ?

RealBasic est un RAD dont l'utilisation est très similaire à Visual Basic. Il est commercialisé par RealSoftware. Il est disponible pour Windows, Mac OS (9 et X) et Linux.

## Quelle est la différence entre la version standard et la version pro de RealBasic ?

La version Pro possède toute les fonctionnalités de la version standard plus :

- la compilation Cross Plateforme qui permet de générer en un seul clic des versions de l'application pour Windows 98 et supérieure, Mac OS 9, Mac OS X et Linux (avec GTK+) ;
- le support des bases de données multi-utilisateurs ;
- la possibilité de créer des applications "consoles" ;
- le support de SSL ;
- un débuggeur cross plateforme distant ;
- le support des ServerSockets pour créer rapidement des applications client/serveur ;
- les Controle Container (voir Qu'est-ce que les Controles Container ? (Lien50));
- la classe AutoDiscovery pour créer des applications qui s'auto-détectent sur le réseau.
- et bien évidemment les prix changent aussi.

## Qu'est-ce que les Controles Container ?

Il s'agit d'un "contrôle de contrôle" : On dispose plusieurs contrôles (boutons, liste, etc.) dans un contrôle Container pour former un super contrôle qui peut être réutilisé à loisir dans le même projet ou dans un nouveau. C'est aussi très utile pour harmoniser une interface.

## RealBasic génère-t-il des applications Universal Binaries pour Mac OS X ?

Oui, il peut compiler en UB, en INTEL seul ou en PPC seul.

## Quelle est la politique de mise à jour de RealSoftware ?

Lors de l'achat de RealBasic vous bénéficiez de 6 mois de mises à jour gratuites. Au delà, il faut souscrire une sorte d'abonnement. Celui-ci peut être souscrit à tout moment, même après la fin de la période initiale comprise avec la licence. RealSoftware publie généralement une nouvelle version tous les 90 jours.

## Puis-je installer RealBasic sur plusieurs machines ?

La licence indique qu'il est possible de l'installer sur deux machines à condition de ne pas les utiliser simultanément.

## Comment exécuter différentes parties de code selon le système sur lequel il s'exécute ?

En utilisant l'opérateur #If... #EndIf et la constante appropriée :

- TargetCarbon pour Mac OS 9
- TargetMachO pour Mac OS X
- TargetMacOS pour Mac OS, que ce soit 9 ou X
- TargetMacOSClassic pour Mac OS 9 tournant sous X, ce qui est appellé Classic
- TargetLinux pour Linux
- TargetWin32 pour Windows

Exemple:

```
#If Target MacOS
     MsgBox("Cette application fonctionne sous Mac OS")
#EndIf
```

## Puis-je réutiliser des projets Visual Basic avec RealBasic ?

Oui car les deux langages sont très similaires. De plus RealSoftware met à disposition un outil : Visual Basic Project Converter. Voir ce tutriel (Lien51).

## Comment faire des calculs facilement ?

En passant par spotlight.
Faites ainsi : POMME + ESPACE (ou COMMAND + ESPACE) (active le champ de saisi de spotlight) puis le calcul à faire.
Le résultat apparaîtra comme résultat de la recherche.

Cela ne fonctionne que sous Léopard

## Quelle est la version de maven intégré dans Leopard ?

C'est la version 2.0.6.

## Quelle est la version de Ruby et Rails livrée avec Léopard ?

Ruby est disponible en version 1.8.6 et Rails en 1.2.3

## L'outil NetInfo a disparu. Comment accéder à ces fonctionnalité sous léopard ?

Dans l'onglet compte, cliquez-droit sur un compte->options avancés. Attention lorsque vous modifiez ces options. Vous pouvez rendre inopérant votre compte.

Un second outil qui existe depuis Panther(au moins) est disponible en ligne de commande. C'est l'outil dscl. Il vous permet de faire toute sorte d'opération sur les comptes et les utilisateurs à la manière des commandes user* et group* sous Linux.

Note : Merci à spoutnik pour la confirmation de l'existence de l'outil dscl.

## Quelle version de Java est livrée avec leopard ?

Leopard est livré non seulement avec la version 5 de Java (J2SE 5.0 | 1.5.0_13-b05) mais aussi la version 1.4 (J2SE 1.4 | 1.4.2_16-b05).

De plus, ces versions sont disponibles aussi bien en 64 bits qu'en 32 bits.

Par défaut, si java est appelé depuis le terminal, c'est la version 32 bits qui est lancée. Il est possible de passer des paramètres -d64 ou -d32 pour indiquer qu'on veut utiliser la JVM 64 bits ou la JVM 32bits.

A noter qu'Apple propose maintenant une version du JDK 6 pour Leopard, mais toujours en Béta.

## Quelles API Java sont livrées avec Leopard ?

- JUNIT 4.1, que l'on retrouve dans le répertoire /usr/share/junit
- Maven 2.0.6 que l'on retrouve dans le répertoire /usr/share/maven
- Ant 1.7 que l'on retrouve dans le répertoire /usr/share/ant

## Quoi de neuf concernant Java dans Xcode 3.0 ?

XCode 3.0 vous propose des templates pour des projets Java qui seront pilotés par des scripts Ant.
Dommage, maintenant que Maven 2.0.6 est livré avec Leopard, que XCode 3.0 ne propose pas de templates pour des projets Java pilotés par Maven.

## Quelle est la version de Ant livrée avec Leopard ?

C'est la version 1.7 de Ant qui est installée avec Leopard

## Comment lire les fichiers odt et openxml ?

**Sur Tiger :**
NeoOffice permet de lire les deux formats.
OpenOffice permet de lire aussi les .odt

**Sur Leopard :**
Il y a NeoOffice,
Mais TextEdit est aussi capable de lire les formats odt et openXML. (Mais les images ne sont pas affichées)

Retrouvez ces Q/R sur la FAQ Mac : Lien52

# Liens

Lien1 : http://t-templier.developpez.com/tutoriel/java/osgi/osgi1/
Lien2 : http://julien-pauli.developpez.com/tutoriels/php/observer-spl/
Lien3 : http://www.456bereastreet.com/archive/200308/css_frames/
Lien4 : http://www.456bereastreet.com/lab/cssframes/
Lien5 : http://www.456bereastreet.com/archive/200411/who_framed_the_web_frames_and_usability/
Lien6 : http://www.quirksmode.org/css/100percheight.html
Lien7 : http://www.w3.org/TR/CSS21/visuren.html#propdef-position
Lien8 : http://css.developpez.com/tutoriels/pseudo-frames/fichiers/exemple_1/
Lien9 : http://css.developpez.com/tutoriels/pseudo-frames/fichiers/exemple_2/
Lien10 : http://css.developpez.com/tutoriels/pseudo-frames/
Lien11 : http://www.la-grange.net/w3c/html4.01/struct/lists.html#h-10.3
Lien12 : http://xhtml.developpez.com/tutoriels/liste-definition/fichiers/exemple1.html
Lien13 : http://xhtml.developpez.com/tutoriels/liste-definition/fichiers/exemple2.html
Lien14 : http://xhtml.developpez.com/tutoriels/liste-definition/fichiers/exemple3.html
Lien15 : http://xhtml.developpez.com/tutoriels/liste-definition/fichiers/exemple4.html
Lien16 : http://xhtml.developpez.com/tutoriels/liste-definition/fichiers/exemple5.html
Lien17 : http://xhtml.developpez.com/tutoriels/liste-definition/fichiers/exemple6.html
Lien18 : http://xhtml.developpez.com/tutoriels/liste-definition/fichiers/exemple7.html
Lien19 : http://xhtml.developpez.com/tutoriels/liste-definition/fichiers/exemple8.html
Lien20 : http://xhtml.developpez.com/tutoriels/liste-definition/fichiers/exemple9.html
Lien21 : http://xhtml.developpez.com/tutoriels/liste-definition/
Lien22 : http://www.wampserver.com/dl.php
Lien23 : http://dcabasson.developpez.com/articles/javascript/ajax/documentation-prototype-1.4.0/
Lien24 : http://openrico.org/downloads
Lien25 : http://j-seignalet.developpez.com/tutoriaux/php-ajax/calendrier/
Lien26 : http://labs.live.com/volta/blog/Announcing+Volta+Web+Development+Using+Only+The+Materials+In+The+Room.aspx
Lien27 : http://www.microsoft.com/downloads/details.aspx?FamilyId=D95598D7-AA6E-4F24-82E3-81570C5384CB&displaylang=en
Lien28 : http://labs.live.com/volta/download/volta-current.exe
Lien29 : http://lgmorand.developpez.com/dotnet/volta/part1/#L3
Lien30 : http://dotnet.developpez.com/livres/
Lien31 : http://www.oracle.com/technology/tech/oci/index.html
Lien32 : http://vicenzo.developpez.com/tutoriels/c/ocilib/
Lien33 : http://blog.developpez.com/index.php?blog=159&title=nokia_s_offre_trolltech_pour_la_nouvelle&more=1&c=1&tb=1&pb=1
Lien34 : http://matthieu-brucher.developpez.com/livre/python/scientifiques/
Lien35 : http://www.swig.org/
Lien36 : http://matthieu-brucher.developpez.com/tutoriels/python/swig-numpy/
Lien37 : http://python.developpez.com/livres/
Lien38 : http://www.developpez.net/forums/forumdisplay.php?f=293
Lien39 : http://thierryaim.developpez.com/tutoriel/innosetup/iscode/
Lien40 : http://thierryaim.developpez.com/tutoriel/innosetup/isdistrib/
Lien41 : http://www.jrsoftware.org/
Lien42 : http://vb.developpez.com/outils/
Lien43 : http://vb.developpez.com/
Lien44 : http://eric-sanchis.developpez.com/linux/shell/bash/
Lien45 : http://blog.developpez.com/index.php?blog=49&title=ecryptfs_encrypter_vos_donnees&more=1&c=1&tb=1&pb=1
Lien46 : http://mac.developpez.com/evenements/MacWorld2008/
Lien47 : http://developer.apple.com/
Lien48 : http://homepage.mac.com/aglee/downloads/appkido.html
Lien49 : http://www.fatcatsoftware.com/plisteditpro/
Lien50 : http://mac.developpez.com/faqs/mac/?page=RealBasic#REALBASIC_ControlesContainer
Lien51 : http://general.developpez.com/realbasic/migration-vb-realbasic/
Lien52 : http://mac.developpez.com/faqs/mac/