Developpez Magazine

Edition de Octobre-Novembre 2007.
Numéro 12.
Magazine en ligne gratuit.
Diffusion de copies conformes à l'original autorisée.
Réalisation : Baptiste Wicht
Rédaction : la rédaction de Developpez
Contact : magazine@redaction-developpez.com

Index

Editorial

Quoi de mieux que de passer l'hiver au coin du feu avec de la lecture ? A défaut de feu, Developpez.com vous offre de la lecture. Découvrez dans ce nouveau magazine, une série d'articles, de critiques de livres, de questions/réponses sur diverses technologies.

La rédaction

Article JavaScript

Ajax avec Prototype

Cet article a pour but de vous présenter les différents moyens que le framework JavaScript Prototype vous offre pour utiliser AJAX.

par **Aurélien Millet**
Page 15

Article PHP

Intégration PHP JasperReports

Ce document a pour but d'expliquer comment mettre en oeuvre l'appel de rapports réalisés au format JasperReports depuis une application PHP via le pont PHP / JAVA Bridge.

par **Charly Clairmont**
Page 10

Introduction à Sitemesh ou le layouting sans douleur

Cet article est un petit tutoriel d'introduction à Sitemesh, un framework Java/J2EE web permettant de gérer facilement le layout d'une application web. Dans cet article, je vais commencer par introduire les autres possibilités de gestion de layout (inclusion de JSP, Tiles), puis je présenterais Sitemesh et en donnerai un petit exemple pour finir sur ses fonctionnalités avancées.

1. Introduction

La gestion des layouts dans les applications web a toujours été pour moi une épine dans le pied jusqu'à ce que je découvre Sitemesh.
Une API de layouting, doit permettre d'éviter de copier/coller du code inutile pour décrire dans chaque page le header, la navigation, le footer, ... C'est ce que permet Sitemesh.

Avant Sitemesh, trois solutions :

- Inclusion de JSP
- Tiles
- Cocoon

L'inclusion de JSP est basique et très limitée car on ne peut réellement tirer profit des framework MVC avec elle. Pour moi ce n'est pas une solution donc je ne vais pas en parler ici.

Je ne connais pas cocoon, bien que j'aie étudié son fonctionnement au moment de choisir le nouveau framework pour gérer le layout. Je pense ce framework très intéressant, en voici une petite introduction (pour plus de détail, le site web français officiel est ici) :
Cocoon est un framework fondé sur la séparation des domaines techniques et de l'assemblage de composants. Il implémente ces concepts par la notion de "pipelines de composants", chaque composant du pipeline étant dédié à une fonction particulière. Cela permet d'utiliser une approche du type "Lego" pour la construction d'applications web, en assemblant des chaînes de composants, sans nécessiter de programmation. Pour cela, on peut par exemple utiliser des transformations XML/XSL directement via Cocoon.

Sitemesh se base sur le pattern décorateur pour gérer le layout des applications web. Mais qu'est-ce donc que cela me direz-vous? (ceux qui connaissent peuvent passer au paragraphe suivant).
Le pattern Décorateur permet de modifier dynamiquement les comportements de certains objets en leur ajoutant de nouvelles fonctionnalités. Le pattern Décorateur fonctionne sur le principe des "Lego": on créé de nouveaux comportements en assemblant des modules, qui constituent les "briques" de notre application. On peut facilement chaîner les décorateurs pour pouvoir ajouter des fonctionnalités complexes grâce à des enchaînements de modules. Les décorateurs permettent aussi de coder des chaînes de traitement, l'idée essentielle est de pouvoir définir un traitement complexe comme un assemblage de traitements simples, offrant ainsi souplesse et modularité. Pour aller plus loin, l'article complet des design patterns de pcaboche est ici avec une bonne explication du design pattern décorateur (dont je me suis inspiré).

2. Quelques mots de Tiles

Très longtemps j'ai utilisé Tiles qui a de nombreux avantages :

- Intégration à Struts et Spring
- Peut être utilisé comme framework seul (ce n'est pas un MVC complet mais ça peut rendre pas mal de service pour une couche web basique)
- Configuration XML permettant l'héritage
- Possibilité de splitter le fichier de configuration en plusieurs parties
- Orientation composant avec un contrôleur par composant
- Bonne performance (un collègue à fait un test avec 1000 tiles imbriqués générés ... et ça n'a pris que quelques centaines de millisecondes à être affiché)

Mais qui, hélas, comprend aussi de nombreux inconvénients :

- Fichier de configuration peu structuré vite complexe et brouillon
- Pas de possibilité de partage de layouts entre différents sites
- Pas de gestion spécifique des headers HTML (voir les possibilités de Sitemesh dans ce domaine)
- Très vite, beaucoup d'héritage entre composants rendent le fichier de configuration très peu lisible
- Très verbeux en configuration
- Le principe de composant amène très vite à créer énormément de composants, donc énormément de JSP et le temps de première compilation d'une page en devient très long
- La gestion des erreurs au niveau d'un composant Tiles n'est pas aisée (impossible de faire une redirection web depuis un contrôleur Tiles car à ce stade, la requête est déjà commité)
- Utilisable uniquement avec des JSP

3. L'approche de Sitemesh

Sitemesh lui, a une approche fort différente: Sitemesh utilise le design pattern décorateur pour ajouter des briques à votre page. Concrètement, vous ne développez que l'intérieur des pages, puis vous définissez dans Sitemesh un layout à utiliser dans lequel vous donnez les différents décorateurs qui vont aller ajouter des parties d'HTML (donc vont aller décorer) vos pages. La page interne que vous allez développer va être calculée, puis en fin de requête le filtre de Sitemesh va décorer votre page selon ce que vous avez configuré. Votre page va être parsée par Sitemesh et va pouvoir ensuite être accessible aux décorateurs. Bien sûr, Sitemesh permet la composition, c'est-à-dire que vous pouvez décorer une page qui est elle-même décorée par une autre ... Mais attention aux effets de bord et aux risques de décoration "en boucle". (Page A, décoré par B, B est lui-même décoré par C, C est lui-même décoré par A ...).

La principale différence de Sitemesh se situe dans le fait que vos pages peuvent être créées avec n'importe quelle technologie. Par exemple, vous définissez une page /toto.do en Spring MVC (ou

en Struts ou n'importe quel framework que vous aimez), vous utilisez ensuite le filtre Servlet de Sitemesh et vous lui dites d'intercepter les requêtes en *.do. Sitemesh, lorsque vous appelez /toto.do, va alors intercepter la requête et la décorer.

Dans Sitemesh, les décorateurs sont eux-mêmes des pages complètes (donc, /footer.do, /header.do, /navigation.do ou /toto.do peuvent être vues indépendamment, en HTML et peuvent toutes êtres des contrôleurs Spring MVC), ce qui permet de les développer et les tester indépendamment. Et en plus, Sitemesh peut décorer une page d'un site, par une page d'un autre site! Pour cela il suffit de définir la page du décorateur comme une page externe (donc commençant par http://), attention, cela peut engendrer des problèmes si vous utilisez de la réécriture d'URL car Sitemesh utilise un HttpClient pour accéder à ces pages (ouvre une session sur le serveur distant, demande la page, puis fait la décoration). Le fait que chaque page, décorée ou "décorante", ait un HTML complet peut ouvrir des horizons et des possibilités multiples, par exemple définir en XHTML une navigation qui pourra être réutilisable indépendamment depuis un site web en Spring MVC via une décoration de Sitemesh ou depuis un composant Flex ou Ajax qui utiliserait la même navigation que vous ... Vos composants de pages deviennent indépendant les uns des autres! Sitemesh n'opérant qu'en fin de requête, n'importe qu'elle technologie web peut être utilisé du côté JAVA. Sitemesh lui-même, pour les pages décorées, permet d'utiliser les technologies JSP, Velocity ou FreeMarker. Par contre, pour les décorateurs, aucune limite, on peut très bien, en plus des technologies JAVA, décorer ses pages par des HTML statiques, des pages PHP, des scripts CGI, ...

Dernière fonctionnalité évoquée : la gestion des headers. Comme les pages décorées et les décorateurs sont des pages HTML complètes, elles contiennent toutes un titre, des CSS,... Sitemesh permet donc, au niveau de la décoration, de spécifier des headers par défaut puis de les réécrire par les valeurs des headers de l'objet décoré. En fait, Sitemesh parse les différentes pages HTML et nous donne accès aux éléments head et body de la page pour pouvoir aller puiser dedans les informations qui nous intéressent.

En plus, le passage de paramètre par les header n'est pas limité aux header de la page finale. On peut très bien définir un paramètre dans les header de la page de contenue qui sera ensuite utilisé dans le body du décorateur. Voici un exemple :

Page décorée
```
<html>
    <meta name="author" content="test@example.com">
    <head>
        <title>Simple Document</title>
    </head>
    <body>
        Hello World! <br />
    </body>
</html>
```

Décorateur
```
 <%@ taglib uri="sitemesh-decorator" prefix="decorator"
%>
 <decorator:usePage id="myPage" />
 <html>
       <head>
              <title>My Site - <decorator:title
default="Welcome!" /></title>
              <decorator:head />
       </head>
       <body>
              <h1><decorator:title default="Welcome!"
/></h1>
              <a href="mailto:<decorator:getProperty
property="meta.author" default="staff@example.com"
/>">
                     <decorator:getProperty
property="meta.author" default="staff@example.com" />
              </a>
             <decorator:body />
       </body>
 </html>
```

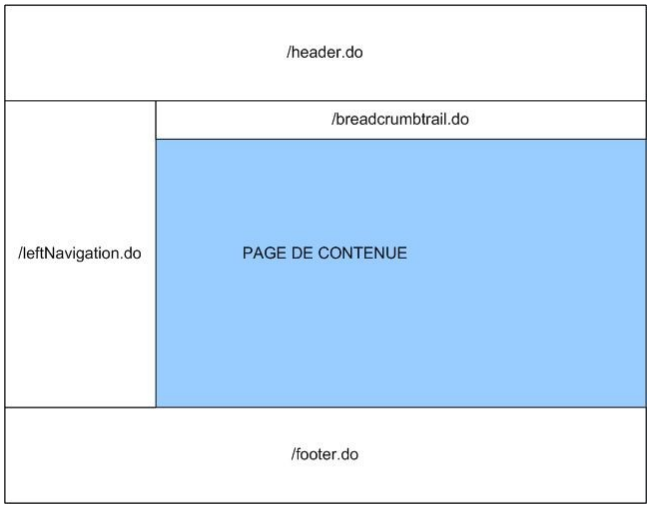
Bien sûr, il existe encore plein d'autres fonctionnalités pour ce framework très simple d'utilisation et de principe, mais très performant. Allez voir sur le site pour plus de détails. Je précise quand même que j'utilise Tiles depuis 3 ans et que je viens de découvrir Sitemesh, que j'utilise pour un nouveau projet, et j'en suis très content alors pourquoi pas vous (ça sonne comme une pub à la télé :=).

Petit récapitulatif Sitemesh:

- Pattern decorator qui permet la séparation entre le code et le layout, l'intégration du layout se faisant via un filtre en fin de requête
- Possibilité d'avoir les éléments du layout dans un site externe (donc de les partager entre plusieurs applications)
- Tous les éléments (éléments de layout et pages internes) sont des HTML complets et donc développables et testables indépendamment
- Agnostique par rapport à la technologie web utilisée (Java - Struts, Spring, ... -, PHP, CGI, ...)
- Agnostique par rapport à la technologie utilisée pour les pages décorées (JSP, Vélocity, Freemarker)
- Gestion intégrée des différents header des pages et des décorateurs
- Nombreuses fonctionnalités avancées (voir plus bas)

4. Exemple

Rappelons tout d'abord, le schéma du site:



Définition du filtre de servlet
```
<filter>
       <filter-name>sitemesh</filter-name>
       <filter-
class>com.opensymphony.module.sitemesh.filter.PageFilt
er</filter-class>
</filter>
```

Définition du fichier de configuration de Sitemesh
```
<?xml version="1.0" encoding="utf-8"?>
<decorators defaultdir="/view/decorator">
   <decorator name="main" page="main.jsp">
```

```
        <pattern>/*</pattern>
        <excludes>
        <pattern>/ErrorPage</pattern>
        </excludes>
    </decorator>
    <decorator name="empty" page="empty.jsp"/>
</decorators>
```

Ici, on définit la JSP main.jsp comme description de layout et on lui dit d'appliquer ce layout à toutes les pages à l'exclusion de la page /ErrorPage. Le décorateur 'empty' servant à pouvoir définir des décorateurs qui ne décorent pas!

La page de layout

```
< ?xml version="1.0" encoding="utf-8"?>
< %@ taglib
uri="http://www.opensymphony.com/sitemesh/decorator"
prefix="decorator" %>
< %@ taglib
uri="http://www.opensymphony.com/sitemesh/page"
prefix="page" %>
< !DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<decorator:usePage id="myPage" />
<html>
    <head>
        <title><decorator:title default="DEFAULT
TITLE"/></decorator></title>
        <decorator:head/>
    </head>
    <body>
        <div><page:applyDecorator name="empty"
page="/decorator/header"/></page></div>
        <div><decorator:body /></div>
        <div><page:applyDecorator name="empty"
page="/decorator/footer"/></page></div>
    </body>
</html>
```

Ici, on définit donc la page HTML globale, on lui précise des éléments du header global qui pourront être redéfinis dans l'élément décoré puis on a ensuite la page HTML de layout elle même.

- *<decorator:title/> : renvoie le titre de la page décorée>*
- *<decorator:head/> : renvoie le header de la page décorée*
- *<decorator:body/> : renvoie le body de la page décorée*
- *<decorator name="empty" page="/decorator/footer"/> : applique le layout donné dans l'attribut 'name' sur la page donnée dans l'attribut 'page' et met le résultat ici. Grâce à cette ligne on décore la page interne par des morceaux d'autres pages, et on remarque que l'on peut chaîner les décorateurs facilement.*
- *<decorator:usePage id="myPage" /> : permet de faire un "binding" sur l'objet page (donc l'objet représentant la page décorée) pour qu'il soit accessible depuis les décorateurs, et donc la page de layout définie ici.*

5. Fonctionalités avancées

5.1. Les DecoratorMapper

En plus du mappage standard par fichier de configuration pour définir quel décorateur utiliser pour quel page. Sitemesh permet d'utiliser d'autre stratégie de mapping. Elles sont bien sûres à définir au sein du fichier de configuration de Sitemesh (sitemesh.xml).

En voici un exemple de configuration :

Configuration d'un PageDecoratorMapper

```
<mapper
class="com.opensymphony.module.sitemesh.mapper.PageDec
oratorMapper">
        <param name="property.1" value="meta.decorator"
/>
</mapper>
```

Les différents mapper de Sitemesh

- PrintableDecoratorMapper: permet de définir un décorateur spécifique pour la version print d'une page
- PageDecoratorMapper: permet de spécifier le nom du décorateur en attribut meta de la page
- ParameterDecoratorMapper: se base du des paramètres de requête pour savoir quel décorateur utiliser
- FrameSetDecoratorMapper: mapper spécifique à l'utilisation des frames
- CookieDecoratorMapper: se base sur la valeur d'un cookie pour choisir quel décorateur utiliser
- RobotDecoratorMapper: permet de définir un décorateur spécifique pour les robots

5.2. Le tag <content>

Le tag content permet de passer des données d'une page JSP à un décorateur. Dans la page, il faut définir un tag content dans lequel on met les données, puis dans la page du décorateur, on utilise le tag decorator:getProperty pour récupérer les données. Exemple ci-dessous :

Définition du tag content dans la page décorée

```
<content tag="pageName">
    Login Page
</content>
```

Récupération des données dans le décorateur

```
<decorator:getProperty property="page.pageName"/>
```

On peut remarquer que Sitemesh utilise un scope pour accéder aux données de la page: meta.property (comme vu précédemment), permet d'accéder aux propriétés des meta tags, et page.property permet d'accéder aux données définies dans la page même.

5.3. Accès avancés aux données de la page décorée (accès à l'objet Page)

Il y a trois manières avancées d'accéder aux données d'une page décorée, ces trois manières fonctionnent aussi bien pour accéder aux données du scope head que page:

1. En utilisant les tags JSP et scriplet

```
<%@ taglib
uri="http://www.opensymphony.com/sitemesh/decorator"
prefix="decorator" %>
<decorator:usePage id="thePage" />
<% String author = thePage.getProperty("meta.author");
%>
```

2. Par Velocity

```
$page.getProperty("meta.author")
```

3. Accès en pure JAVA

```
import com.opensymphony.module.sitemesh.Page;
import
com.opensymphony.module.sitemesh.RequestConstants;
...
Page thePage =
request.getAttribute(RequestConstants.PAGE);
```

```
String author = thePage.getProperty("meta.author");
```

Je n'ai hélas pas réussit à faire fonctionner ce dernier sous WebSphere 5.1, l'objet Page obtenue est null. Mais normalement, cela est sensé fonctionner

D'un point de vue logique, si l'accès est possible via un scriplet, il doit aussi être possible via une expression EL, bien que non testé ça donnerait le code suivant

4. En utilisant les tags JSP et EL (non testé)

```
<%@ taglib
uri="http://www.opensymphony.com/sitemesh/decorator"
prefix="decorator" %>
<decorator:usePage id="thePage" />
${thePage.property["meta.author"]}
```

Tout cela étant équivalent bien sur à l'écriture utilisée à la section III :

Ecriture standard grâce à la taglib

```
<decorator:getProperty property="meta.author" />
```

5.4. Accéder à la requête de la page décorée depuis un décorateur

L'objet Page contient la requête, en y accédant, on a donc accès à l'objet requête de la page décorée. Sitemesh faisant l'assemblage de la page décorée et des éléments de décorations, dans un décorateur, la requête n'est pas celle de la page décorée. L'accès à la requête se fait en JAVA.

Accéder à la requête décorée

```
import com.opensymphony.module.sitemesh.Page;
import
com.opensymphony.module.sitemesh.RequestConstants;
...
Page thePage =
request.getAttribute(RequestConstants.PAGE);
HttpServletRequest decoratedRequest =
thePage.getRequest()
```

6. Liens

- Site officiel de Sitemesh : Lien1
- Un article très complet, surtout sur les DecoratorMapper : Lien2
- La FAQ officielle : Lien3
- L'article initial sur mon blog : Lien4
- Article sur le pattern decorator : Lien5

Retrouvez l'article de Loïc Mathieu en ligne : Lien6

JNI pour les Nuls... ou JNA pour faire plus simple !

"Write once, run anywhere" : le slogan de Java a toujours mis en avant la portabilité du langage et de ses APIs, en promettant qu'un même code pourra être exécuter n'importe où. Et si on peut dire que cela est globalement vrai, ce n'est pas toujours un avantage !

En effet l'API se trouve ainsi dépourvu de certaine fonctionnalité qui peuvent sembler "basique" sur un système, mais qui ne sont pas forcément disponible sur d'autres. Et même si Java à récemment mis de l'eau dans son vin en incorporant des fonctionnalités "optionnelles" ou au fonctionnement dépendant du système hôte, il reste toujours nécessaire de s'attaquer au code natif dès que l'on s'approche un peu trop du système...

Au grand malheur des développeurs qui voient **JNI** comme une usine à gaz, qui utiliserait un bazooka pour tuer une mouche...

... et ils n'ont pas complètement tord !

Car si **JNI** est assez puissant et remplit bien son rôle, il souffre de plusieurs défauts qui le rendent assez complexe à mettre en œuvre. Il existe pourtant une alternative assez attirante, bien qu'elle ne soit pas standard : **JNA**.

Mais voyons cela un peu plus en détails...

1. Avant propos...

Nous allons prendre un exemple tout simple : on va appeler depuis une application Java une fonction native présente dans une librairie native dynamique. Pour l'exemple on utilisera la fonction **strcmp()** bien que son intérêt soit totalement limité.

L'objectif étant plutôt de se concentrer sur le travail que cela représente...

1.1. Avant propos sur le chargement des librairies dynamiques

Par défaut, **Java** respecte les conventions du système hôte pour le chargement des librairies natives, c'est à dire :

- Sous **Windows**, les librairies seront recherchées dans le **PATH**.
- Sous **Unix/Linux**, elles sont recherchées dans le **LD_LIBRARY_PATH**.
- Sous **Mac OS**, c'est la variable d'environnement **DYLD_LIBRARY_PATH** qui est utilisée.

Il est possible d'outre-passer cela en modifiant la variable système **java.library.path** (ou **jna.library.path** pour **JNA** que nous verrons un peu plus loin). Si la librairie ne fait pas partie d'un des répertoires spécifiés, l'exécution du programme génèrera une **UnsatisfiedLinkError**...

De même, chaque système possède ses propres conventions pour le nommage des fichiers représentant les librairies, par exemple pour une librairie nommé "hello" :

- Sous **Windows**, on lui ajoute simplement l'extension **.dll**, soit **hello.dll**.
- Sous **Unix/Linux**, on utilise le prefixe **lib** couplé à l'extension **.so**, soit **libhello.so**.
- Sous **Mac OS**, on utilise le prefixe **lib** couplé à l'extension **.jnilib**, soit **libhello.jnilib**.

2. Amusons nous avec JNI

Le principal défaut de JNI vient du fait qu'il n'est pas possible d'appeler n'importe quelle fonction native directement : il est ainsi obligatoire de définir une méthode native qui respectent un prototype bien précis. Ainsi on est obligé de passer par une méthode intermédiaire qui englobera cet appel.

2.1. Déclarer la méthode native

La première étape est toute simple et consiste donc à écrire le prototype Java de la méthode native, par exemple on pourrait avoir la classe suivante :

```
package jnidemo;

public class JNIDemo {
  public native int strcmp(String s1, String s2);
}
```

Ce code peut être compilé normalement sans problème puisque le compilateur ne vérifie pas les liens vers les méthodes natives, par contre l'exécution génèrera une belle exception puisque la méthode native correspondante n'existe pas encore...

2.2. Générer le header C/C++

Une fois cette classe compilé, il faut utiliser l'outil **javah** (fourni avec le JDK) afin de générer un fichier d'entête **C/C++**. Ce dernier s'utilise comme la commande **java** et neccessite donc un nom de classe complet (c'est à dire avec le package) :

```
javah jnidemo.JNIDemo
```

Ce qui nous génèrera dans le cas présent un fichier nommé "jnidemo_JNIDemo.h" et contenant le code suivant :

```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include <jni.h>
/* Header for class jnidemo_JNIDemo */

#ifndef _Included_jnidemo_JNIDemo
#define _Included_jnidemo_JNIDemo
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:     jnidemo_JNIDemo
 * Method:    strcmp
 * Signature: (Ljava/lang/String;Ljava/lang/String;)I
 */
JNIEXPORT jint JNICALL Java_jnidemo_JNIDemo_strcmp
  (JNIEnv *, jobject, jstring, jstring);

#ifdef __cplusplus
}
#endif
#endif
```

2.3. Implémenter le code natif

Il est maintenant nécessaire de coder la fonction native correspondant au prototype généré, ce qui pourrait donner en C :

```
#include <string.h>     /* pour strcmp() */
#include "jnidemo_JNIDemo.h"

JNIEXPORT jint JNICALL Java_jnidemo_JNIDemo_strcmp (
  JNIEnv* env,     /* Environnement JNI */
  jobject thiz,    /* Pointeur "this" de l'instance
courante */
  jstring s1,      /* Argument #1 */
  jstring s2       /* Argument #2 */
) {
  /* On "transforme" les chaines Java en chaines C : */
  const char* str1 = (*env)->GetStringUTFChars(env, s1,
0);
  const char* str2 = (*env)->GetStringUTFChars(env, s2,
0);

  /* On appelle la fonction strcmp() : */
  int result = strcmp(str1, str2);

  /* On libère la mémoire utilisée pour les chaines C :
*/
  (*env)->ReleaseStringUTFChars(env, s1, str1);
  (*env)->ReleaseStringUTFChars(env, s2, str2);

  /* Et enfin on retourne le résultat : */
  return result;
}
```

Premier constat : le code natif est assez lourd, puisqu'il nécessite des conversions de types de Java vers C et inversement ainsi qu'une gestion des allocations mémoires (puisqu'on sort du cadre d'utilisation du GC). Bref pour un simple appel de fonction on se retrouve dans un nids de guêpes...

2.4. Compiler et générer la librairie native

Il nous faut désormais compiler ce bout de code. Pour cela il faut spécifier au compilateur l'emplacement des headers natif de JNI, qui se trouvent dans le répertoire include du JDK, ce qui nous donne (la variable d'environnement JAVA_HOME pointant vers le chemin d'installation du JDK) :

Ce qui donne sous Linux avec gcc :

```
gcc -I $JAVA_HOME/include -I $JAVA_HOME/include/linux
-c jnidemo_JNIDemo.c
```

Et l'équivalent sous Windows avec mingw :

```
gcc -I %JAVA_HOME%\include -I
%JAVA_HOME%\include\win32 -c jnidemo_JNIDemo.c
```

On peut enfin générer notre librairie native, que l'on nommera "compare", ce qui donne pour Linux :

```
gcc -shared jnidemo_JNIDemo.o -o libcompare.so
```

Et pour Windows :

```
gcc -shared -Wl,--kill-at jnidemo_JNIDemo.o -o
compare.dll
```

L'option supplémentaire **-Wl,--kill-at** indique au linkeur qu'il ne doit pas modifier le nom des fonctions exportées (qui sont "décorées" par défaut), car cela semble causer des problèmes à JNI.

2.5. Charger la librairie native

Il reste une petite modification à effectuer sur le code source de notre classe Java : il est impératif de charger cette librairie pendant le chargement de la classe afin que la méthode native puisse être utilisée sans problème. Pour cela il suffit d'ajouter un bloc static dans le corps de la classe :

```
package jnidemo;

public class JNIDemo {

  /* Bloc static : le code est exécuté une seule fois
   * lors du chargement de la classe
   */
  static {
    /* On charge la librairie en utilisant son nom de
base : */
    System.loadLibrary("compare");
  }

  public native int strcmp(String s1, String s2);
```

```
}
```

On peut désormais utiliser notre méthode native de manière tout à fait standard :

```
package jnidemo;

public class Main {

  public static void main(String[] args) {

    if (args.length != 2) {
      System.err.println("Paramètre absent !");
      System.exit(1);
    }

    JNIDemo demo = new JNIDemo();

    System.out.printf("strcmp('%s', '%s') => %d %n",
      args[0], args[1], demo.strcmp(args[0], args[1]) );

  }
}
```

Le constat est assez rapide pour moi : Tout ça pour ça !
Pour un simple appel de méthode, on se retrouve à suivre un mode d'emploi en cinq étapes :

1. Déclarer une méthode native
2. Générer le header C/C++
3. Implémenter le code natif
4. Compiler et générer la librairie native
5. Charger la librairie native

Tout ceci est d'autant plus rageant lorsqu'on se contente d'appeler une fonction existante comme dans le cas présent, et que ce type de code tient sur quelques lignes dans n'importe quel langage natif (et généralement totalement transparent). Sans compter que l'on devra générer et déployer une librairie par système supporté.

Tout cela alors que le code natif nécessaire pour JNI se contente de faire des conversions de type et des allocations/libérations de mémoire afin de pouvoir appeler la fonction native. Bref rien de très intéressant à coder, mais une source de problème potentiellement...

3. Et pour faire plus simple ?

JNA se présente heureusement comme une alternative beaucoup plus simple d'accès, en permettant d'accéder dynamiquement à n'importe quelle bibliothèque partagée du système sans utiliser JNI (pas directement en tout cas). En fait il s'agit d'une librairie Java/native qui se chargera du chargement des librairies, de l'appel des fonctions et de la conversion des types... si bien qu'il n'y a quasiment rien à faire.

3.1. Déclarer les méthodes natives dans une interface

Contrairement à JNI, la marche à suivre est assez différentes puisqu'il ne faut pas marquer les méthodes avec le mot-clef native, et qu'il est impératif d'utiliser une interface qui contiendra les définitions des fonctions natives (et seulement celles-ci). A l'exécution on récupérera une instance valide de notre interface qui sera automatiquement associé aux méthodes natives correspondante.

Il suffit donc de déclarer toutes les fonctions dans une interface particulière, ce qui donne dans notre cas :

```
package jnademo;

import com.sun.jna.Library;

public interface JNADemo extends Library {

  public int strcmp(String s1, String s2);

}
```

La seule condition est d'étendre l'interface com.sun.jna.Library qui fait simplement office de marqueur...

3.2. Instancier dynamiquement notre interface

Il ne reste plus qu'à créer une instance de cette interface qui sera automatiquement lié à la librairie native. Pour cela il suffit d'utiliser la méthode Native.loadLibrary() en lui précisant le type Java de l'interface et le nom de la librairie dynamique native.

Le seul problème vient du fait que le nom de la librairie peut être différent d'un système à l'autre. Dans ce cas précis la fonction strcmp() fait partit de la librairie "c" sous Unix/Linux, alors qu'il faut utiliser "msvcrt" sous Windows...

Mais il n'y a rien de bien méchant puisqu'il suffit de vérifier le nom du système pour régler le problème :

```
  // On détermine le nom de la librairie selon le
système :
  String libName = "c";
  if
(System.getProperty("os.name").contains("Windows")) {
    libName = "msvcrt";
  }
  // On charge la librairie dynamique en l'associant
avec l'interface :
  JNADemo demo = (JNADemo) Native.loadLibrary(libName,
JNADemo.class);
```

Et ? C'est tout ou presque. En effet il suffit ensuite d'utiliser l'instance ainsi créée pour appeler les fonctions natives.

Le programme équivalent deviendrait alors :

```
package jnademo;

import com.sun.jna.Native;

public class Main {

  public static void main(String[] args) {

    if (args.length != 2) {
      System.err.println("Paramètre absent !");
      System.exit(1);
    }

    // On détermine le nom de la librairie selon le
système :
    String libName = "c";
    if
(System.getProperty("os.name").contains("Windows")) {
      libName = "msvcrt";
    }
    // On charge la librairie dynamique en l'associant
avec l'interface :
    JNADemo demo = (JNADemo)
Native.loadLibrary(libName, JNADemo.class);

    System.out.printf("strcmp('%s', '%s') => %d %n",
```

```
        args[0], args[1], demo.strcmp(args[0], args[1]) );

    }
}
```

La librairie s'occupe elle-même de faire toutes les conversions de type et de rechercher les fonctions natives a appeler dynamiquement selon la définition de la méthode Java, si bien qu'il n'y a pas besoin d'écrire une seule ligne de code native !

Même si je n'ai fait que survoler les possibilités qu'offre JNA, et bien qu'il n'y ait qu'une documentation succincte pour le moment, je dois dire que cela me semble vraiment très complet, car cela inclut entre autres :

- Mapping Java/natif automatique des types primitifs et des String.
- Mapping des struct et des union vers des types Java spécifiques (Structure et Union).
- Mapping des pointeurs vers un type Java (ByReference).
- Mapping des pointeurs de fonctions (ou callback) en utilisant une interface Java.
- Possibilité de définir un mapping personnalisé pour ses propres objets Java.
- Mapping automatique de la méthode Java vers la fonction native du même nom, mais en gardant la possibilité d'utiliser une classe qui se chargera de cela (par exemple pour utiliser des noms de méthodes Java différent afin de respecter les règles de nommages Java).
- Gestion des librairies Win32 qui utilise la convention d'appel __stdcall.

Le seule reproche que je pourrais faire, c'est que l'API n'utilise pas les "nouveautés" de Java 5.0, car je pense que les annotations se serait bien appliqué dans ce cas précis...

Pour plus d'information sur le sujet vous pouvez consulter les liens suivants :

- Java Native Interface, la documentation officielle : Lien7
- Java Native Access, la page du projet sur java.net, et sa javadoc online : Lien8

Retrouvez le billet de Frédéric Martini en ligne : Lien9

La FAQ Général Java

A quoi sert l'introspection ou la réflexivité, et comment l'utiliser ?

L'introspection consiste en la découverte dynamique des informations propres à une classe Java ou à un objet. Ce mécanisme est notamment utilisé au niveau de la machine virtuelle Java lors de l'exécution de votre programme, et donne lieu à une API.

Le paquetage java.lang.reflect permet l'introspection en rendant possible l'accès aux classes, à leurs champs, méthodes ou encore constructeurs, et à toutes les informations les caractérisant, même celles qu'on pensaient inaccessibles Elle est également très utile pour instancier des classes de manière dynamique, dans le processus de sérialisation d'un bean Java, ainsi que dans la génération de code.

Les méta données présentes dans les fichiers binaires renseignent sur le contenu de chaque classe répertoriée et permettent à la jvm de procéder à des vérifications lors de l'exécution d'un programme (pensez à l'exception java.lang.NoSuchMethodError).
La jvm utilise également ces informations pour vous proposer la complétion de code dans les environnements de développement Java, ce qui se fait en temps réel (pas de génération à faire au préalable).

Des détails sur le format des méta données dans un fichier binaire sont disponibles dans les documents de spécification de la machine virtuelle Java. Sachez seulement que les champs et méthodes sont identifiés par leur nom, le type (identifiants spécifiques à la jvm) pour les champs, et la signature pour les méthodes. A partir de ces informations, la jvm sait directement localiser la portion de byte code correspondant à l'implémentation d'une méthode.

Comment connaître l'ensemble des classes dont hérite une classe ?

Il existe dans la classe Class une méthode nommée getSuperClass(), c'est cette méthode que nous allons utiliser.

```
Class c = Class.forName("maClasse");
while((c=c.getSuperclass()) != null)
{
   System.out.println(c.getName());
}
```

Comment connaître l'ensemble des interfaces qu'implémente une classe ?

Il existe dans la classe Class une méthode nommée getInterfaces() qui renvoie un tableau des interfaces implémentées par la classe.

```
Class c = Class.forName("maClasse");
Class[] interfaces = c.getInterfaces();
for(int i=0;i<interfaces.length;++i)
{
   System.out.println(interfaces[i].getName());
}
```

Comment accéder dynamiquement à la valeur d'un champ donné d'un objet ?

Pour consulter ou modifier un champ donné d'un objet de façon dynamique, il faut commencer par récupérer l'objet de type Field correspondant au champ en question. Il suffit ensuite d'appeler la méthode correspondante avec pour premier paramètre l'objet cible.
Prenons l'exemple suivant où nous modifions le contenu du champ défini par la variable nomChamp de l'objet obj en lui donnant la valeur définie par la variable val.

```
void changeValeur(Object obj, String nomChamp, Object
val) throws Exception
{
  java.lang.reflect.Field f =
obj.getClass().getField(nomChamp);
   f.set(obj,val);
}
```

Un exemple de consultation de la valeur d'un champ donné :

```
void afficheValeur(Object obj, String nomChamp) throws
Exception
{
   Field f = obj.getClass().getField(nomChamp);
   System.out.println(f.get(obj));
}
```

Remarque : les methodes set et get sont des méthodes générales mais il existe aussi des équivalents pour les types classiques : setDouble(Object obj, double d) ou setBoolean(Object obj, boolean z).

Comment lancer dynamiquement une méthode donnée d'un objet ?

Nous allons utiliser la méthode invoke définie dans la classe Method :

```
Object invoke(Object obj, Object[] args)
```

Voici un exemple générique de lancement dynamique d'une méthode donnée sur un objet :

```
Object lancerMethode(Object obj, Object[] args, String
nomMethode) throws Exception
{
   Class[] paramTypes = null;
   if(args != null)
   {
      paramTypes = new Class[args.length];
      for(int i=0;i<args.length;++i)
      {
         paramTypes[i] = args[i].getClass();
      }
   }
   Method m =
obj.getClass().getMethod(nomMethode,paramTypes);
   return m.invoke(obj,args);
}
```

[Java 5.0] Comment connaitre les annotations d'un élément ?

Java 5.0 permet de marquer certains éléments du langage avec des Annotations, ces dernières peuvent être accessibles lors de l'exécution (seulement pour les annotations dont la rétention est **RetentionPolicy.RUNTIME**). Le package **java.lang.reflect** se voit ainsi doté d'une nouvelle interface qui décrit quatre méthodes permettant d'accéder aux annotations, on y trouve ainsi les méthodes suivantes :

- **getAnnotation(Class)** qui permet d'obtenir une annotation particulière (si elle est présente).
- **getAnnotations()** qui permet d'obtenir un tableau contenant toutes les annotations de l'élément.
- **getDeclaredAnnotations()** qui permet d'obtenir un tableau contenant toutes les annotations directement déclarées sur l'élément (en ignorant ainsi les annotations héritées de la classe parent).
- **isAnnotationPresent(Class)** qui permet simplement de savoir si une annotation particulière est présente.

Cette interface, nommée **AnnotatedElement**, est implémentée par les classes suivantes : **Class**, **Package**, **Constructor**, **Method** et **Field**. Enfin les classes **Constructor** et **Method** proposent également une méthode **getParameterAnnotations()** afin d'accéder aux annotations de leurs paramètres...

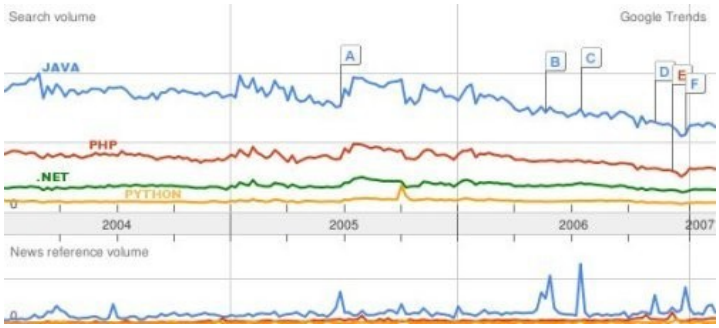
Retrouvez ces questions et de nombreuses autres sur la FAQ Général Java : Lien10

Intégration PHP / JasperReports

Ce document a pour but d'expliquer comment mettre en oeuvre l'appel de rapports réalisés au format JasperReports depuis une application PHP via le pont PHP / JAVA Bridge.

1. Introduction

PHP a gagné d'années en années une image de langage professionnel, simple et performant. Comme le montre l'Association Française des Utilisateurs de PHP dans son livre blanc « PHP en Entreprise » (1), il y a un très grand nombre de sites internet de renom, de sociétés du CAC 40 qui développent en PHP. Ce qui renforce alors cette image de robustesse. Cette notoriété du langage est aussi visible compte tenu du nombre d'articles qui lui est consacré, et le nombre très important de recherches effectuées sur Google(2).



Il existe quelques librairies de bon niveau pour la production de graphiques, de documents au format pdf. Toutefois, peu d'entre elles fournissent un environnement pour assister les concepteurs de rapports dans leur élaboration et leur publication. Agatha Reports (3), un projet Brésilien présente l'ambition d'apporter à PHP les outils de reporting qui lui manque. Cette solution bien que fonctionnelle ne supporte pas à ce jour PHP5 (4). De même, la nouvelle plate-forme de Zend en version 3.0.2 (5) affiche un support pour BIRT(6), un environnement de conception de rapport basé sous Eclipse et appuyé par Actuate.

Jasper Reports a offert au monde Java TM un moteur de grande qualité pour la génération de rapports rassemblant tableaux, graphiques, mêlant des sources de données très hétérogènes (rapport multi-sources). Cet outil en plus d'être très performant sait exporter vers divers formats tels que : pdf, html, xls, xml, cvs, txt, rtf et notamment un support pour Open Office depuis sa dernière version. Par ailleurs, iReport, un environnement de développement pour Jasper Reports procure une assistance de haut niveau dans la construction des autres rapports ou états. En plus, les deux projets suivent une évolution synchronisée. Ainsi, toutes les fonctionnalités ajoutées au moteur Jasper Reports sont prises en compte par iReport.

Nombreuses sont les interrogations postées sur les forums sur les outils de reporting en PHP, ou encore sur l'intégration entre PHP et des environnement de reporting tierce. Voilà une réponse : « PHP et Jasper Reports » ! Celle-ci nous semble élégante parce qu'elle apporte de la productivité, grâce à iReport qui est assez simple à prendre en main évitant alors d'apprendre l'API de la librairie. Ensuite, Jasper Reports est un moteur aujourd'hui éprouvé : de nombreuses solutions libres ou propriétaires l'embarquent, des grandes sociétés comme Siemens, des administrations françaises en font un usage réguliers. Enfin, l'association PHP, Java, via PHP / Java Bridge (7) commence à gagner ses gallons. Même si certains pensent qu'il existe des marges d'amélioration (8). Ce projet rend possible l'instanciation d'objets java depuis des programmes PHP avec la syntaxe PHP.

Vous l'avez compris pour intégrer des rapports Jasper Reports à vos programmes PHP il est nécessaire d'installer PHP / Java Bridge.

2. Installation de PHP / Java Bridge

2.1. installation sous GNU Linux

L'installation sous GNU Linux est assez simple car les packages sont disponibles au format rpm. Il convient de télécharger le fichier php-java-bridge-x.y.z-1-i386.rpm et de l'installer :

fichier rpm pour l'installation du bridge entre php et java
```
rpm -i php-java-bridge-x.y.z-1-i386.rpm
```

où "x.y.z" correspond à la dernière version stable.

Bien sûr il est nécessaire d'avoir une machine virtuelle Java installée sur sa machine. Dans le fichier php.ini il faut au moins initier la variable java.java. Vous devez dans votre fichier php.ini (ou php.d/java.ini) ajouter les entrées :

configuration du fichier php.ini
```
java.java_home={Répertoire d'installation JAVA}
java.java={Répertoire d'installation JAVA}/java
```

où {Répertoire d'installation JAVA} est à remplacer par le chemin vers votre intallation du SDK.

Si vous souhaitez faire dialoguer vos applications PHP et un serveur J2EE installer le package php-java-bridge-tomcat :

fichier rpm pour exploiter une interface avec tomcat
```
rpm -i php-java-bridge-tomcat-x.y.z-1.i386.rpm
```

Pour télécharger les fichiers adéquats voir : Lien11

2.2. Installation sous Windows

Ici, il faut télécharger php-java-bridge_y.x.z_j2ee.zip, le décompresser dans un répertoire temporaire. Parmi les fichiers décompressés il y a une archive web JavaBridge.war. Décompressez le aussi et copiez les fichiers JavaBridge.jar et java-x86-windows.dll respectivement depuis WEB-INF/lib et

WEB-INF/cgi dans le répertoire de vos extension php. Enfin, il vous faut ajouter quelques entrées dans votre fichier php.ini.

configuration php.ini
```
Extension=php_java.dll
```

Votre fichier php.ini intègre alors la ligne précédante et est proche de ce qui suit :

configuration php.ini
```
; windows Extensions
; Note that ODBC support is built in, so no dll is
nedded for it.
; Note that mny DLL files are located in the
extensions/ (PHP 4) ext/ (PHP 5)
...
...
;extension-php_mbstring.dll
;extension-php_bz2.dll
...
...
extension-php_java.dll
...
...
```

Enfin configurez le connecteur Java pour PHP

configuration php.ini
```
;;;;;;;;;;;;;;;;;;;
; Module Settings ;
;;;;;;;;;;;;;;;;;;;

[java]
java.java_home = "c:\Program File\Java\jdk1.5.0_10\bin"
java.java = "c:\Program
File\Java\jdk1.5.0_10\bin\javaw.exe"
java.class.path = "c:\php\ext\JavaBridge.jar"
java.library.path = "c:\php\ext"
;java.hosts = "127.0.0.1:8080"
;java.servlet = On
java.log_level = 2
```

Bien sûr les différentes variables sont à initialiser avec les valeurs relatives à votre evironnement Java (le répertoire où est installé votre SDK), ou de votre serveur d'application J2EE.

pour plus d'iformations : Lien12

2.3. vérification du support java de PHP

Vous devez redémarrer votre serveur apache et vérifier que vous avez bien le support java de PHP.

Créez une page phpinfo.php et insérez y les lignes suivantes :

phpinfo()
```
<?php
	phpinfo();
?>
```

Sauvez ce fichier, publiez le sur votre environnement web, et appelez la page depuis votre navigateur. Vous devrez observer le support java de php.

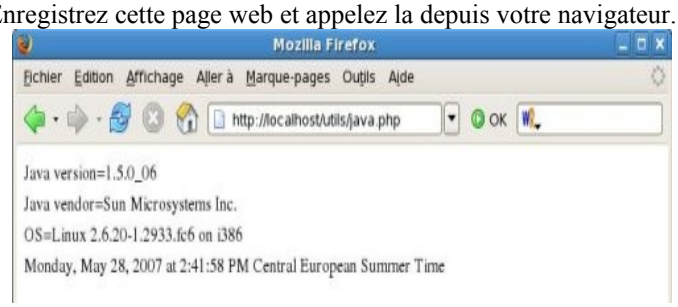
java

java support	Enabled
java bridge	4.0.8
java.java_home	/usr/local/java/jdk1.5.0_06
java.java	/usr/local/java/jdk1.5.0_06/bin/java
java.log_file	<stderr>
java.log_level	no value (use back-end's default level)
java.persistent_connections	On
java.security_policy	Off
java command	JAVA_HOME=/usr/local/java/jdk1.5.0_06 LD_LIBRARY_PATH=/usr/lib/php/modules: /usr/local/java/jdk1.5.0_06/bin/java -Djava.library.path=/usr/lib/php/modules -Djava.class.path=/usr/lib/php/modules/JavaBridge.jar -Djava.awt.headless=true -Dphp.java.bridge.base=/usr/lib/php/modules php.java.bridge.Standalone LOCAL@java-bridge-9f5 1
java status	running
java server	@java-bridge-9f5

II-D. votre première intégration entre PHP et Java
Créez un fichier java.php et collez les lignes suivantes :

appel de java depuis php
```
<?php
	// créer une instance de la classe Java
java.lang.System dans PHP
	$system = new Java('java.lang.System');

	// accéder aux propriétés
	echo 'Java version=' . $system-
>getProperty('java.version') . ' <br />';
	echo 'Java vendor=' . $system-
>getProperty('java.vendor') . '<br />';
	echo 'OS=' . $system->getProperty('os.name') . '
' .
		$system->getProperty('os.version')
. ' on ' .
		$system->getProperty('os.arch') .
'<br />';

	// Exemple avec java.util.Date
	$formater = new
Java('java.text.SimpleDateFormat',
			"EEEE, MMMM dd, yyyy 'at'
h:mm:ss a zzzz");

	echo $formater->format(new
Java('java.util.Date'));
?>
```

Enregistrez cette page web et appelez la depuis votre navigateur.



3. Appel des fichiers JasperReports depuis PHP

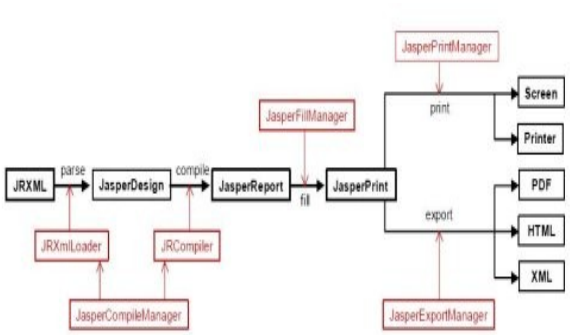
3.1. bâtir votre modèle de document avec iReport

Grâce à iReport vous allez pouvoir construire votre rapport :

- définissez votre connexion à votre base de données ou autre source de données
- créez un nouveau document
- créez une requête avec l'assistant
- Ajoutez les éléments que vous souhaitez à votre rapport
- Enregistrez votre rapport
- Exécutez le rapport.

3.2. écrire le script PHP qui appelle votre fichier JRXML

Dans l'exemple qui suit nous allons procéder à des instanciations des classes de l'API java de Jasper Reports en suivant les étapes généralement indiquées dans les exemples fournis par leur démo.



Créez un fichier que vous pouvez par exemple nommer jasperreports.php et ajoutez les lignes suivantes :

JasperReports depuis un script php

```
<?php
	$reportsPath
="/home/ccharly/publichtml/utils/reports/";
	$reportFileName = "CommandesClients1";
	$jasperReportsLib =
"/home/ccharly/publichtml/utils/jasperlib";

	if(extension_loaded('java')) {
		// lecture du répertoire où sont rengés
les librairies utiles à JasperReports
		$handle = @opendir($jasperReportsLib);

		// ajout de tous les fichier jar au
chemin de classe (Class Path)
		while(($new_item =
readdir($handle))!==false) {
			$java_library_path .=
'file:'.$jasperReportsLib.'/'.$new_item .';';
		}

		try {
			// chargement des librairies au
classpath

			java_require($java_library_path);

			// création de la connexion JDBC
			$Conn = new
Java("org.altic.jasperReports.JdbcConnection");
			// driver
			$Conn-
>setDriver("com.mysql.jdbc.Driver");
			// url de connexion
			$Conn-
>setConnectString("jdbc:mysql://localhost/erpmart");
			// utilisateur
			$Conn->setUser("root");
			// mot de passe
			$Conn->setPassword(null);

			// Compilation du fichier JRXML
en fichier Jasper
			$sJcm = new
JavaClass("net.sf.jasperreports.engine.JasperCompileMan
ager");
			$report = $sJcm-
>compileReport($reportsPath .$reportFileName.".jrxml");

			// Remplir le modèle avec les
données
			$sJfm = new
JavaClass("net.sf.jasperreports.engine.JasperFillManage
r");
			$print = $sJfm->fillReport(
			$report,
			new Java("java.util.HashMap"),
			$Conn->getConnection()
			);

			// Export du fichier au format
pdf
			$sJem = new
JavaClass("net.sf.jasperreports.engine.JasperExportMana
ger");
			$sJem-
>exportReportToPdfFile($print, $reportsPath
.$reportFileName.".pdf");

			if (file_exists($reportsPath
.$reportFileName.".pdf")){
				header('Content-
disposition: attachment;
filename="'.$reportFileName.'.pdf"');
				header('Content-Type:
application/pdf');
				header('Content-Transfer-
Encoding: binary');
				header('Content-Length:
'. @filesize($reportsPath . $reportFileName.".pdf"));
				header('Pragma: no-
cache');
				header('Cache-Control:
must-revalidate, post-check=0, pre-check=0');
				header('Expires: 0');
				 set_time_limit(0);
				@readfile($reportsPath
.$reportFileName.".pdf") or die("problem occurs.");
			}
		} catch (JavaException $ex) {
			$trace = new
Java("java.io.ByteArrayOutputStream");
			$ex->printStackTrace(new
Java("java.io.PrintStream", $trace));
			print "java stack trace:
$trace\n";
		}
	}
?>
```

Vous devez adapter le fichier en modifiant les variables suivantes :

- **$reportsPath** : chemin où sont rangés vos rapports au format jrxml
- **$reportFileName** : nom du fichier à compiler et exporter en pdf (remarque : ici seule la racine du nom du fichier est nécessaire)
- **$jasperReportsLib** : répertoire des librairies nécessaires à l'utilisation de JasperReports. Ce répertoire contient par exemple les mêmes librairies que ceux contenu dans le répertoire lib d'iReport.

La commande java_require charge dans le classpath toutes ces librairies. Veillez à ce que le répertoire désigné par $jasperReportsLib soit accessible à l'utilisateur qui fait tourner apache ainsi que les fichiers qu'il contient.

Ajoutez le fichier alticJasper.jar il contient un petit utilitaire pour créer une connexion JDBC.

classe de la connexion JDBC

```
package org.altic.jasperReports;

import java.io.FileInputStream;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

/**
 * @author ccharly
 *
 */
public class JdbcConnection {
	private String driver;		//
"oracle.jdbc.driver.OracleDriver";
	private String connectString; //
"jdbc:oracle:thin:@YOUR_ORACLE_HOST:1521:YOUR_SID";
	private String user;
	//"YOUR_ORACLE_USER_NAME";
	private String password;		//
"YOUR_ORACLE_PASSWORD";

	public JdbcConnection(){
		//loadPropertiesConnection();
	}

	public JdbcConnection(String driver, String
connectString, String user, String password){
		this.driver = driver;
		this.connectString = connectString;
		this.user = user;
		this.password = password;
	}

	public void loadPropertiesConnection(){
		this.driver = "";
		this.connectString = "";
		this.user = "";
		this.password = "";

		try {
			Properties props = new
Properties();
			InputStream resourceAsStream =
getClass().getResourceAsStream("/connection.properties"
);
			props.load(resourceAsStream);
			this.setDriver(
props.getProperty("driver"));
			this.setConnectString(
props.getProperty("connectString"));

	this.setUser(props.getProperty("user"));
			this.setPassword(
props.getProperty("password"));
		}catch(Exception e){
			e.printStackTrace();
		}
	}

	public Connection getConnection(){
		try {
			//Change these settings according
to your local configuration

			Class.forName(this.getDriver());
			Connection conn =
DriverManager.getConnection(
				this.getConnectString(),
				this.getUser(),
				this.getPassword());
			return conn;
		}catch(ClassNotFoundException e){
			e.printStackTrace();
		}catch(SQLException e){
			e.printStackTrace();
		}
		return null;
	}

	public String toString(){
		return " Driver : " + this.getDriver() +
" | " +
				" ConnectString : " +
this.getConnectString()  + " | " +
				" User : " +
this.getUser()  + " | " +
				" Password : " +
this.getPassword();
	}

	public String getConnectString() {
		return connectString;
	}

	public void setConnectString(String
connectString) {
		this.connectString = connectString;
	}

	public String getDriver() {
		return driver;
	}

	public void setDriver(String driver) {
		this.driver = driver;
	}

	public String getPassword() {
		return password;
	}

	public void setPassword(String password) {
		this.password = password;
	}

	public String getUser() {
		return user;
	}

	public void setUser(String user) {
		this.user = user;
	}
}
```

Vous devez donc modifier les paramètres de connexions vers votre base de données. Pensez aussi à ajouter la librairie du connecteur JDBC vers votre base de données.

Indiquez le driver JDBC de votre base de données, par exemple pour MySql

configuratioin d'une connexion JDBC depuis PHP - 1
```
$Conn->setDriver("com.mysql.jdbc.Driver");
```

Indiquez l'url de connexion

configuratioin d'une connexion JDBC depuis PHP - 2
```
$Conn->setConnectString("jdbc:mysql://{Nom du
S}:{port}/{Nom de la Base de données}");
```

Indiquez le nom de l'utilisateur

configuratioin d'une connexion JDBC depuis PHP - 3
```
$Conn->setUser({utilisateur};
```

Indiquez le mot de passe

configuratioin d'une connexion JDBC depuis PHP - 4
```
$Conn->setPassword({Mot de passe});
```

Une fois ces paramétrages terminés, vous êtes en mesure de générer vos rapports JasperReports depuis PHP. La compilation n'est sans doute pas utile si vous publier directement les fichiers « .jasper » que vous pouvez récupérer depuis iReport.

4. Conclusion

Ce petit exemple montre comment il est ainsi possible d'exploiter JasperReports via une application PHP. Cette intégration apporte à PHP un outil de reporting performant et particulièrement productif grâce à iReport. Les concepteurs de rapports se retrouve dans un environnement convivial et productif.

Pour augmenter à ce titre la productivité il serait intéressant d'encapsuler JasperReports afin d'en masquer la complexité pour ne pas rebuter les utilisateurs de PHP déjà habitués à une syntaxe relativement simple. Prenons par exemple la connexion vers la base de données il faudrait une ligne de code qui serait :

simplifier connexion JDBC - 1
```
$conn->getMySqlConnection({nom du Serveur}, {Port},
{Base de données}, {Utilisateur}, {Mot de passe});
```

ou bien

simplifier connexion JDBC - 2
```
$conn->getConnection({Type de Base},{nom du Serveur},
{Port}, {Base de données}, {Utilisateur}, {Mot de
passe});
```

où {Type de Base} vaudrait : ORACLE, MYSQL, POSTGRESQL

L'encapsulation éviterait aussi aux développeurs PHP d'avoir à manipuler les appels directs aux classes java, ce qui en simplifierait la lecture du code.

Le projet Dynamic Jasper offre une simplification de l'API JasperReports. Il serait intéressant de se baser sur ce projet pour développer une librairie de scripts PHP pour JasperReports.

5. Pour aller plus loin

- PHP / Java Bridge Lien13
- Jasper Reports and PHP C'est un très bon article qui apporte déjà une certaine abstraction de la complexité de Jasper Report en PHP. Il faut je pense aller encore plus loin. Mais c'est un très bon début. Lien14
- Projet Jasper Reports Lien15
- Projet iReport http://jasperforge.org/sf/projects/ireport
- PHP et BEA Weblogic Lien16
- Les cours sur les générateurs d'états - sur Developpez.com Lien17
- Intégration de FOP et PHP / Java Bridge Lien18
- Dynamic Jasper Lien19

(1) Livre Blanc "PHP en entreprise" - Lien20
(2) Lien21
(3) Lien22
(4) Lien23
(5) Lien24
(6) Lien25 - Lien26
(7) Lien27
(8) Lien28

Retrouvez le tutoriel de Charly Clairmont en ligne : Lien29

Se simplifier AJAX avec Prototype

En un peu plus de deux ans, le terme "AJAX" s'est répandu comme une traînée de poudre dans les bouches des développeurs Web. Interactivité, rapidité, tels sont les mots d'ordre des utilisateurs au sujet des applications. AJAX est là pour y répondre. Et comme un bonheur n'arrive jamais seul, des outils apparaissent pour aider les développeurs. Cet article a pour but de vous présenter les différents moyens que le framework JavaScript Prototype vous offre pour utiliser AJAX.

1. Introduction à AJAX et Prototype

1.1. Communication synchrone

Une application Web est un dialogue client / serveur. Le serveur est en écoute et attend que les clients viennent lui demander un service. Un serveur HTTP répond aux requêtes HTTP. Il passe à différents programmes les paramètres envoyés par le client. Ces programmes retournent du code HTML au serveur qui termine la transaction en faisant suivre ce code au client, dans une réponse HTTP.

Jusqu'à il y a peu (les technologies du Web évoluent très vite), les applications Web étaient synchrones. C'est-à-dire que l'utilisateur effectue une action sur une page Web, son navigateur envoie la requête au serveur, interprète sa réponse et charge la nouvelle page correspondant au code reçu. Pendant tout cet échange, l'utilisateur a lâché sa souris pour attendre sagement de voir apparaître ce qu'il a demandé.

1.2. Communication asynchrone

Aujourd'hui, il est possible de passer outre ce fonctionnement par l'utilisation toute récente de ce qui est désigné par l'acronyme bien connu : AJAX, soit : Asynchronous JavaScript And XML.
Alors, AJAX, qu'est-ce que c'est donc ?! Déjà, ce n'est pas UNE technologie. C'est l'utilisation conjointe de différentes technologies. Ensuite, on ne peut pas réellement dire que c'est une NOUVELLE technologie. Tout dépend du point de vue en fait. Les éléments qui lui servent de base (HTML, DOM, JavaScript, XML et objet XmlHttpRequest) lui sont bien antérieurs. Mais c'est l'utilisation de ces éléments qui est nouvelle.

Le pivot central est l'objet XmlHttpRequest (XHR). C'est le portage par Mozilla de l'objet XmlHttp développé par Microsoft et intégré à Internet Explorer 5 en tant qu'objet ActiveX. Cet objet permet d'envoyer des requêtes HTTP et de récupérer les données retournées par le serveur. Au fur et à mesure de l'avancée du traitement de la requête HTTP côté client puis côté serveur, l'objet passe par différents états. A chaque changement d'état, il déclenche un événement. Il est donc possible de suivre facilement la progression du traitement.

Il s'agit toujours d'échanges HTTP entre un client et un serveur mais cet échange est transparent pour l'utilisateur. Il n'attend plus forcément le chargement d'une nouvelle page en restant inactif. Son navigateur attend une réponse mais l'utilisateur, lui, a toujours la main. Classiquement, AJAX est utilisé pour mettre à jour dynamiquement une partie d'une page Web.

1.3. AJAX : intérêts et inconvénients

Le principal intérêt d'AJAX est d'apporter beaucoup plus d'interactivité aux applications Web. L'utilisateur n'a plus besoin de valider une page entière pour que ses actions soient prises en compte, il peut modifier l'interface à chaque clic. On évite ainsi le problème de la page blanche qui est affichée pendant quelques secondes le temps de charger toutes les données. Cette page blanche peut parfois dérouter l'utilisateur inexpérimenté.

L'autre avantage de réduire le volume des données transférées à chaque appel est que cela réduit le temps nécessaire au serveur pour répondre. On peut donc avoir des réactions quasi instantanées aux actions de l'utilisateur.

Un autre avantage est de décharger le serveur d'une partie du travail. Il collecte les informations demandées et les renvoie au client. A ce dernier de savoir quoi en faire, grâce à du code JavaScript. Son pendant négatif direct est la nécessité de disposer d'un navigateur Web supportant le JavaScript. Sans cela, impossible de naviguer correctement.

Il peut aussi y avoir un problème pour le développeur. Utiliser les technologies AJAX rajoute du code JavaScript et sans cadre précis, ce code finira par être brouillon et difficilement maintenable.

Attention aussi aux abus ! A trop chercher l'interactivité, on perd en accessibilité. AJAX peut également soulever des questions au niveau de la sécurité et du référencement.

1.4. Prototype : de l'utilité d'un framework

Voilà ma définition d'un framework : ensemble de bibliothèques / classes / fonctions et de conventions permettant le développement rapide d'applications. Donne un cadre structuré et impose une rigueur entraînant la production de code fiable et facile à maintenir. Vous pouvez aussi consulter la définition de framework dans le dictionnaire de Developpez.com.

Etant un développeur CakePHP convaincu, l'intérêt d'utiliser ce genre de "cadres de travail" me paraît juste évident. Aux débuts du Web, développer ses pages statiques en HTML dans son coin, ça passait. Aujourd'hui on parle dynamique et on parle surtout efficace, rapide, sûr. Non aux développements à l'artisanal au fond du garage pour un code cochon. Dites oui aux frameworks ! Vive le code lisible ! Aimez-vous les uns les autres. Pardon...

Le JavaScript est généralement le parent pauvre du développement Web. Ce n'est évidemment pas une règle générale mais le code est souvent réalisé rapidement, pour répondre aux besoins sans chercher à affiner, tant que ça marche. Et il faut

tester sur tous les navigateurs puisqu'aucun ne supporte le même standard. Prototype (Lien30) c'est le chevalier en armure rutilante qui vient sur son fier destrier blanc immaculé sauver le pauvre développeur de ces horreurs. Il vous offre les moyens d'écrire du code concis et clair. Se chargeant des problèmes de navigateurs et autres tâches pénibles, il vous laisse vous concentrer sur l'aspect métier, ce que doit faire votre code. Et pour ça, on lui dit un grand merci !

1.5. AJAX sans et avec Prototype

Sans cadre spécial pour utiliser AJAX, il faut disposer d'une fonction spécifique qui va d'abord instancier un objet XmlHttpRequest, de différentes façons suivant le navigateur utilisé. Elle doit ensuite associer une fonction aux changements d'états de l'objet XHR (appelons la xhrStateChange). Elle doit enfin ouvrir une connexion avec le serveur en spécifiant une URL et éventuellement des paramètres.

La main passe ensuite à la fonction que nous avons nommée xhrStateChange. Elle est appelée à chaque changement d'état de l'objet XHR. On peut lui demander d'exécuter du code suivant l'état qui lui est indiqué. La transaction est considérée terminée avec succès lorsque xhr.readyState vaut 4 et lorsque xhr.status vaut 200 (ou plus généralement quand il est inférieur à 400). Quand c'est le cas, il faut ensuite passer la réponse du serveur à une fonction qui va se charger de mettre à jour la page Web avec ces nouvelles informations.

Vous pouvez regarder le code proposé par siddh dans cet article (Lien31) pour voir ce qu'il est nécessaire de mettre en oeuvre dans ce type de fonctionnement.

Avec Prototype, plus besoin de vous embêter à savoir si votre script crée correctement l'objet XHR ou si vous attendez bien le bon état. Vous créez un objet, auquel vous indiquez la page à interroger, les éventuels paramètres et la fonction à appeler quand la réponse a été reçue. C'est un exemple simple, il est possible de préciser plus de choses mais nous verrons les détails plus tard. Une ligne peut donc suffire à gérer la requête asynchrone, sans avoir à se préoccuper des problèmes liés à la plateforme d'exécution du JavaScript !

A noter que pour des raisons de sécurité, les requêtes AJAX ne peuvent être faites que vers des URLs de même domaine que la page contenant le JavaScript en cours d'exécution (c'est-à-dire même protocole, même serveur et même port). Ceci vaut pour AJAX en général, Prototype n'a rien à voir là-dedans. Pour passer outre il faut par exemple passer par un "proxy" en PHP (lit le document demandé et en retourne le contenu) ou demander à l'utilisateur de diminuer les paramètres de sécurité de son navigateur.

2. Envoyer une requête

Prototype permet d'utiliser trois objets pour gérer des requêtes asynchrones :

- Request Ajax.Request(url[, options]) Instancie un objet XHR, envoie la requête au serveur et reçoit sa réponse.
- Updater Ajax.Updater(container, url[, options]) Effectue les mêmes opérations que Request puis met à jour le contenu d'un élément de la page avec les données reçues.
- PeriodicalUpdater Ajax.PeriodicalUpdater(container, url[, options]) Effectue les mêmes opérations que PeriodicalUpdater mais répétées à intervalles réguliers.

Comme on peut le voir ci-dessus, ces trois objets ont tous au moins 2 paramètres en commun :

- url : Adresse de la page à interroger.
- options : Paramètres de la requête et fonctions callback. Ce paramètre est optionnel.

Remarque :
L'émission d'une requête HTTP via AJAX se fait suite à la levée d'un événement JavaScript. Je ne détaillerai pas cette partie mais juste au passage, petit conseil pour ceux qui gèrent encore leurs évènements JavaScript directement dans le code XHTML (ex : <a href="..." onclick="javascript:...) : laissez tomber cette méthode ! Utilisez plutôt les gestionnaires d'évènements de Prototype. Cela séparera définitivement les codes XHTML et JavaScript, c'est plus propre et plus facile à maintenir.
Pour en savoir plus sur ce sujet : la documentation officielle (Lien32) ou la traduction sur developpez.com (v1.4.0) (Lien33). Attardez-vous surtout sur la méthode Event.observe().

2.1. Options configurables

L'un des paramètres (optionnel) est donc une liste d'options. Elle est passée sous forme d'un élément au format JSON (assimilable à un tableau associatif). Voici les différentes options accessibles et les valeurs qu'il est possible d'indiquer :

Nom	Valeur par défaut	Description
asynchronous	true	Indique si la requête est synchrone ou non.
contentType	application/x-www-form-urlencoded	En-tête "Content-Type" de la requête. Par défaut, indique que c'est une URL qui est envoyée. A modifier pour envoyer par exemple du XML.
encoding	UTF-8	Encodage du contenu de la requête.
method	POST	Méthode HTTP de la requête. Autre valeur possible : "GET".
parameters	null	Paramètres de la requête, à passer au script appelé. Ils peuvent être exprimés principalement sous deux formes : {a: 1, b: 5} ou 'a=1&b=5'
postBody	none	Contenu spécifique pour le corps de la requête. Si une valeur est indiquée, l'argument "parameters" ne sera pas pris en compte.
requestHeaders	X-Requested-With : 'XMLHttpRequest' X-Prototype-Version : version courante de Prototype Accept : 'text/javascript,	En-têtes de la requête. Peut recevoir une valeur sous deux formes : un objet (ses attributs donnent les

	text/html, application/xml, text/xml, */*' Content-type : construit en fonction de contentType et encoding	valeurs des en-têtes) ou un tableau (case paire : nom de l'en-tête ; case impaire : valeur).

2.2. Fonctions callbacks

Les callbacks représentent différents points du cycle d'exécution d'une requête. Il est possible d'associer du code à ces étapes. Pour la liste complète des callbacks : la documentation officielle ou la traduction sur developpez.com (v1.4.0). Il faut être prudent en les utilisant car certains sont implémentés différemment suivant les navigateurs.

Les deux callbacks qui sont le plus couramment utilisés sont "onSuccess" et "onFailure". Les deux sont invoqués quand une requête est terminée. onSuccess si le code d'état de la requête est entre 200 et 299, onFailure sinon.
Ils reçoivent deux paramètres :
- L'objet XHR (souvent appelé "transport") ;
- L'évaluation JSON de la réponse si réception d'un en-tête X-JSON, null sinon.

2.3. L'objet Request

- Ajax.Request(url[, options])
- Objet de base pour traiter les requêtes AJAX : instancie un objet XHR, envoie la requête au serveur et reçoit sa réponse. Il vous laisse indiquer quoi demander et quoi faire de la réponse.
- Paramètres : URL de la page à interroger et liste optionnelle des options.

Exemple d'utilisation de l'objet Request

```
new Ajax.Request(
	'http://mon-domaine.com/ma/page.php',
	{
		method: 'get',
		parameters: {nom1: valeur1, nom2: valeur2},
	}
);
```

Voir une page exemple : Lien34

2.4. L'objet Updater

- Ajax.Updater(container, url[, options])
- Effectue les mêmes opérations que Request puis met à jour le contenu d'un élément de la page avec les données TEXTE reçues (celles contenues dans responseText).
- Paramètres : identifiant de l'élément DOM à mettre à jour, URL de la page à interroger et liste optionnelle des options. En plus des options décrites plus haut, l'objet Updater en propose deux autres :

Nom	Valeur par défaut	Description
evalScripts	false	Le contenu de la réponse peut contenir des balises <script>. Ce paramètre détermine si ces éléments sont évalués ou pas. Si evalScripts est à "true", ces blocs ne seront pas inclus dans la page mais passés à la fonction eval(). Si vous définissez des fonctions dans ces blocs, il ne faut pas utiliser "function foo() { ... }" mais "foo = function() { ... }".
insertion	none	Par défaut, quand un élément DOM est mis à jour, son ancien contenu est effacé. Si ce comportement n'est pas souhaité, il est possible d'indiquer où ajouter, dans l'élément, le texte de la réponse. Les valeurs possibles sont : "Insertion.After", "Insertion.Before", "Insertion.Bottom", "Insertion.Top".

Il est possible de mettre à jour un élément DOM suivant l'état de la réponse :
Updater('monDiv'...) : c'est toujours l'élément monDiv qui est concerné ;
Updater({ success: 'monDiv' }...) : l'élément monDiv n'est mis à jour qu'en cas de réussite de la requête ;
Updater({ success: 'monDiv', failure: 'monAutreDiv' }...) : le texte de la réponse sera affiché dans l'élément monDiv en cas de réussite ou dans l'élément monAutreDiv en cas d'échec.

Exemple d'utilisation de l'objet Updater

```
new Ajax.Updater(
	'monDiv',
	'http://mon-domaine.com/ma/page.php',
	{
		method: 'get',
		parameters: {nom1: valeur1, nom2: valeur2},
		insertion: Insertion.Bottom
	}
);
```

Voir une page exemple : Lien35

2.5. L'objet PeriodicalUpdater

- Ajax.PeriodicalUpdater(container, url[, options])
- Effectue les mêmes opérations que PeriodicalUpdater mais répétées à intervalles réguliers. Comme Updater, ne prend en compte que le contenu de responseText.
- Paramètres : identifiant de l'élément DOM à mettre à jour, URL de la page à interroger et liste optionnelle des options. PeriodicalUpdater supporte les options de base ainsi que celles apportées par Updater (se référer aux sections dédiées). Il en propose également deux nouvelles :

Nom	Valeur par défaut	Description
frequency	2	Temps en secondes entre deux exécutions de requêtes (c'est en fait une période et non une fréquence). Ne pas mettre un temps trop court au risque d'envoyer des requêtes avant que les précédentes ne se soient terminées.
decay	1	Multiplicateur de la période quand la réponse du serveur ne change pas. A chaque fois que la réponse reçue est la même que la précédente, la valeur courante de "frequency" est multipliée par la valeur de "decay". Les requêtes seront donc envoyées de plus en plus

		espacées. Quand la réponse reçue diffère de la précédente, "frequency" retrouve sa valeur d'origine. La valeur par défaut de "decay" est 1, ce qui veut dire que "frequency" n'est pas modifiée.

Remarque : il est possible d'arrêter l'exécution d'un objet PeriodicalUpdater avec la méthode stop(). Pour le relancer, utiliser start().

Exemple d'utilisation de l'objet PeriodicalUpdater

```
new Ajax.PeriodicalUpdater(
	'monDiv',
	'http://mon-domaine.com/ma/page.php',
	{
		insertion: Insertion.Bottom,
		frequency: 5
	}
);
```

Voir une page exemple : Lien37

3. Recevoir et traiter une réponse

Réagir au changement d'état d'un objet XHR se fait en associant du code à un état. Mais il y a deux façons de faire cela :

- Spécifiquement pour une requête donnée ;
- Globalement pour tous les objets XHR qui seront créés, via l'utilisation d'objets "responders".

3.1. Comportements spécifiques

C'est la méthode la plus utilisée : pour une requête donnée, on attend un certain type de réponse qui doit subir un traitement précis. Pour cela, lors de la déclaration de la requête, il faut préciser, dans les paramètres, les états auxquels réagir (les callbacks) et leur associer du code à exécuter.

Exemple : on informe l'utilisateur quand la requête est terminée

```
new Ajax.Request(
	'http://mon-domaine.com/ma/page.php',
	{
		method: 'get',
		parameters: {nom1: valeur1, nom2: valeur2},
		onSuccess: function() { alert('Requête terminée avec succès.') },
		onFailure: function() { alert('Requête échouée.') }
	}
);
```

3.2. Comportements génériques ("responders")

Si pour un même état de tous les objets XHR, un même code doit être exécuté, il est possible de ne pas l'écrire dans les déclarations de toutes les requêtes. Il suffit de l'enregistrer en tant que "responder".

Exemple : alerter à chaque fois qu'une exception est levée

```
Ajax.Responders.register({
	onException: function() { alert('Une exception a été levée !') };
});
```

Il est aussi possible de supprimer ce qui a été enregistré mais pour cela il faut avoir gardé une référence.

Exemple : alerter à chaque fois qu'une exception est levée

```
responder = function() { alert('Une exception a été levée !') };
Ajax.Responders.register({
	onException: responder
});

Ajax.Responders.unregister(responder);
```

3.3. Pré-requis aux sections suivantes

- **Comme déjà mentionné dans la partie sur les callbacks (section 4.2), les fonctions associées au traitement de la réponse reçoivent deux paramètres :**
- L'objet XHR (souvent appelé "transport") ;
- L'évaluation json de la réponse si réception d'un en-tête X-JSON, null sinon.

Dans les parties suivantes nous prendrons comme exemple l'instanciation suivante :

```
new Ajax.Request(
	'server.php',
    {
        onSuccess: function(transport, json) {
            document.write(
                "=> transport.responseText : " + transport.responseText
                + "<br />=> transport.responseXML : " + transport.responseXML
                + "<br />=> json : " + json
            );
        }
    }
);
```

3.4. Recevoir du texte

Si aucun en-tête particulier n'est émis avec la réponse, le flux reçu est stocké dans l'attribut responseText de l'objet XHR.

server.php

```
<?php echo 'Ceci est un test d\'envoi de simple texte.' ?>
```

Résultat affiché

```
=> transport.responseText : Ceci est un test d'envoi de simple texte.
=> transport.responseXML : null
=> json : null
```

3.5. Recevoir du XML

Si par contre un en-tête XML est émis en début de flux de réponse, son contenu sera accessible via les attributs responseText et responseXML.

server.php

```
<?php
header("Content-type: text/xml");
echo '<?xml version="1.0" encoding="UTF-8"?>';
?>
<root>
	<data>Donnee 1</data>
	<data>
		<sub1>Donnee 2</sub1>
		<sub2>Donnee 3</sub2>
	</data>
```

```
</root>
```

Résultat affiché

```
=> transport.responseText :  Donnee 1   Donnee 2
Donnee 3
=> transport.responseXML : [object XMLDocument]
=> json : null
```

responseText contient sous forme de texte l'ensemble de la réponse (balises et données). Les balises ne sont pas affichées dans notre exemple puisque le navigateur a essayé de les interpréter.
responseXML contient la réponse sous forme de flux XML et peut être manipulé comme tel.

3.6. Recevoir du JSON

Le JSON (JavaScript Object Notation) (Lien37) est plus simple, plus lisible et plus léger que le XML. Comme en plus, Prototype en facilite l'évaluation, il est conseillé de l'utiliser quand c'est possible. Si un en-tête X-JSON est émis en début de flux de réponse, Prototype l'analyse automatiquement et stocke le résultat dans l'objet json.
Il est possible de générer du JSON en PHP, soit avec l'implémentation native depuis PHP 5.2 soit en utilisant cette librairie (Lien38).

server.php (avec la fonction native de PHP)

```php
<?php
$datas = array(
    'root' => array(
        'data' => 'Donnee 1',
        'otherData' => array(
            'sub1' => 'Donnee 2',
            'sub2' => 'Donnee 3'
        )
    )
);

header("X-JSON: " . json_encode($datas));
?>
```

server.php (en utilisant de la librairie)

```php
<?php
$datas = array(
    'root' => array(
        'data' => 'Donnee 1',
        'otherData' => array(
            'sub1' => 'Donnee 2',
            'sub2' => 'Donnee 3'
        )
    )
);

require_once('JSON.php');
$json = new Services_JSON();
header("X-JSON: " . $json->encode($datas));
?>
```

Résultat affiché

```
=> transport.responseText :
=> transport.responseXML : null
=> json : [object Object]
```

L'objet json est alors facilement exploitable. Dans notre exemple, pour accéder à la valeur de "sub1", il suffit de passer par "json.root.otherData.sub1".

4. Conclusion

J'espère que cet article vous aura montré les avantages énormes qu'apporte l'utilisation de Prototype, ne serait-ce que sur la gestion d'AJAX. Sachant que ce n'est qu'une petite partie de ce qu'il propose, je vous recommande très fortement son utilisation pour accélérer vos développements JavaScript et simplifier votre code.
Vous pouvez entre autres lire le code des pages utilisées comme exemples au cours de cet article. J'y utilise différents aspects de Prototype (gestion d'évènements, manipulation d'éléments, méthodes utiles comme $ et $$...).

Liens sur Developpez.com

- Section AJAX : FAQ, cours, tutoriels... : Lien39
- Traduction de la documentation Prototype v1.4.0 : Lien40
- AJAX et l'objet XMLHttpRequest : Lien41

Retrouvez l'article de Aurélien Millet en ligne : Lien42

Au coeur des dictionnaires en .Net 2.0

1. Introduction

Le Framework 2.0 compte parmi ses classes une grande panoplie de collections aussi diverses l'une de l'autre et conçues pour un besoin spécifique. Les développeurs .Net se trouvent fréquemment dans une situation où ils ne savent pas laquelle des collections choisir et se tournent généralement vers la plus basique d'entre elles pour arriver à leurs fins. Ce choix rapide et maladroit entraine souvent le ralentissement de l'algorithme à implémenter et aussi des effets de bords incompréhensibles et difficiles à cerner si on ne s'aperçoit pas que la collection choisie est en fait la cause des malheurs.

Le but de cet article, est d'assister le développeur à choisir le bon dictionnaire selon le cas de figure dans lequel il se trouve. Je vais traiter tout au long de cet article les avantages et les limites de chacun des dictionnaires du Framework tout en me focalisant sur les méthodes les plus intéressantes qu'offrent chacune des classes.

2. Présentation générale

Du point de vue organisationnel, les collections sont groupées en 3 namespaces :

- **System.Collections** : ce namespace regroupe un ensemble de collections non typées permettant de stocker des données hétérogènes dans la même structure dynamique puisque la signature des méthodes accepte le type object.
- **System.Collections.Generics** : La grande nouveauté du Framework 2.0, les generics reprennent essentiellement les collections du premier namespace en prenant en charge des données typées.
- **System.Collections.Specialized** : comme son nom l'indique, ce namespace regroupe un ensemble de collections spécialisées qui répondent à un besoin spécifique pour un type de donnée spécifique. Contrairement aux collections du namespace System.Collections, celles du namespace Specialized sont fortement typées.

Les collections qui seront traitées dans cet article sont les suivantes :

- **System.Collections**
  - Hashtable
- **System.Collections.Specialized**
  - OrderedDictionary
  - StringDictionary
  - NameValueCollection
  - ListDictionary
  - HybridDictionary
- **System.Collections.Generics**
  - Dictionnary
  - SortedDictionnay
  - System.Collection.ObjectModel
  - KeyedCollection

Du point de vue logique, je regrouperais les collections du Framework en 5 classes. Bien entendu une collection du Framework peut appartenir à plusieurs groupes.

- **Les collections basiques** : Ce sont les collections à comportement basique. Elles permettent de stocker des données dynamiquement dans une structure de données en se limitant à des fonctionnalités de base telles que l'ajout, la suppression, l'énumération, etc.
- **Les listes séquentielles** : Ces collections suivent une certaine logique d'ordre lors de l'insertion des données. Il existe deux sortes de collectons séquentielles dans le Framework 2.0 ; la pile (LIFO : Last In First Out) et la file (FIFO : First In First Out).
- **Les dictionnaires** : Ce sont des structures dynamiques qui permettent d'emmagasiner des données sous forme de clé/valeur afin de rendre instantanée la récupération d'une valeur sachant sa clé.
- **Les collections spécialisées** : Ce sont des structures spéciales qui représentent des comportements spécifiques applicables généralement à un type donné. Par exemple une structure qui représente une suite binaire ou une structure qui n'accepte que des types strings.
- **Les generics** : C'est l'ensemble des collections fortement typées. Elles font parties systématiquement du namespace System.Collections.Generics.

3. La Complexité des algorithmes

Afin de comparer l'efficacité des différentes collections du Framework, nous aurons besoin d'une norme qui permettrait d'évaluer le coût d'exécution des opérations offertes par les collections. La théorie de la complexité algorithmique permet justement d'évaluer la rapidité d'exécution des tâches algorithmiques (méthodes de classes). Tout au long de cet article la complexité dans le pire des cas a été adoptée. Elle permet en fait d'évaluer les algorithmes dans le cas où les paramètres d'entrées engendrent le maximum d'itérations pour converger vers une solution.

Prenons l'exemple d'une liste chainée représentée par la structure suivante :

5	4	42	2	8	36	9	47

Dans le meilleur des cas, l'algorithme de la méthode GetAt, qui permet de récupérer une valeur à partir de son index, converge en 1 itération. Ce cas est représenté par l'appel GetAt(0). On dit que le l'algorithme dans le meilleur des cas est en O(1).

Dans le pire des cas, l'algorithme de la méthode GetAt converge en 8 itérations. Ce cas est représenté par l'appel GetAt(7). Pour généraliser, on dit que l'algorithme dans le pire des cas est en O(n), n représente la taille des entrées (taille du tableau dans notre cas).

La théorie de la complexité algorithmique utilise la notation Landau (O : grand o) pour représenter l'efficacité. En faisant

abstraction des explications mathématiques, voici les différents types de complexités que l'on va rencontrer dans cet article classés du plus rapide au plus lent :

- **O(1)** : complexité constante indépendante des paramètres d'entrées.
- **O(log(n))** : complexité logarithmique, le temps d'exécution croit légèrement par rapport à la taille des entrées.
- **O(n)** : complexité proportionnelle à la longueur de l'entrée.

4. Les dictionnaires dans le Framework 2.0

4.1. Introduction aux dictionnaires

Les dictionnaires représentent des structures de données particulières qui permettent d'associer chaque valeur à une clé. La particularité de ce mapping est que la recherche d'une valeur par sa clé est presque instantanée et converge vers une complexité en O(1) même si dans certains cas elle peut monter jusqu'en O(n) si la fonction de hachage est mal implémentée. Cette rapidité de recherche est possible grâce aux tables de hachages sur lesquelles reposent les dictionnaires. La question qui se pose donc est : comment cette structure permet de retrouver les valeurs quasi instantanément.

En fait, une table de hachages permet d'associer à chaque clé, quelque soit son type, un entier calculé grâce à une fonction. La valeur entière représente généralement l'index de la clé dans la table et la fonction en question est appelée la fonction de hachage.

Concrètement, dans le Framework 2.0, la méthode GetHashCode représente la fonction de hachage. Le Framework dispose de l'implémentation de cette fonction pour quelques types comme le string. L'utilisation de string comme clé assure donc la performance de recherche dans la table de hachage puisque le Framework assure l'unicité de la valeur de hachage pour chaque combinaison de caractères stockée dans un string. Cette unicité empêche le cas où deux clés retournent le même code de hachage de se produire. Ce phénomène est appelé collision et est responsable de la perte de performance de la table de hachage. La méthode GetHashCode n'est pas tout le temps implémentée au niveau du Framework, certainement pas dans le cas où on veut utiliser des classes personnalisées comme clé. Dans ce cas, le développeur prend en charge l'implémentation de la fonction de hachage en surchargeant la méthode GetHashCode et en implémentant l'interface IEquatable. Ce sujet sera traité plus en détail dans la suite de l'article.

note msdn : Par exemple, l'implémentation de la méthode GetHashCode fournie par la classe String retourne des codes de hachage uniques pour des valeurs de chaîne uniques. Par conséquent, deux objets String retournent le même code de hachage s'ils représentent la même valeur de chaîne.

Prenons l'exemple du cas où on veut stocker dans un dictionnaire les noms des pilotes de formule 1 à qui on veut associer leurs totaux de points au cours de la saison.

```
Dictionary<string, int> lesPilotes = new
Dictionary<string, int>(2);

lesPilotes.Add("Lewis Hamilton", 70);
lesPilotes.Add("Fernando Alonso", 68);

foreach (string piloteKey in lesPilotes.Keys){
    //afficher les clés et leurs codes de hachage
    Console.WriteLine("Clé = {0}, HashCode = {1}",
    piloteKey, piloteKey.GetHashCode());
}

//Clé = Lewis Hamilton,  HashCode = 710750954
//Clé = Fernando Alonso, HashCode = 477124616
```

Nous remarquons que les deux clés retournent des codes de hachages différents qui serviront à retrouver directement le score correspondant au nom du pilote.

4.2. Dictionary

La classe Dictionary et la plus basique des génériques dans le contexte des structures en clé/valeur. Elle correspond à la classe hashtable (table de hachage) des collections de base. Nous allons voir dans ce qui suit les possibilités d'insertion et de suppression dans la classe Dictionary et son comportement après ces opérations.

Quand faut-il choisir la classe Dictionary ?

1. Collection typée qui stocke des listes de clé/valeur ; l'identification des clés similaires se fait en implémentant, la classe qui représente la clé, l'interface IEquatable. Il est possible aussi de passer une classe qui implémente IEqualityComparer lors de la construction de la collection.
2. N'autorise pas les clés doublons.
3. Rapide : accès et suppression en O(1)

Manipulation d'un dictionnaire

```
//Initialiser la taille pour gagner en performance.
Dictionary<string, string> lesPilotes = new
Dictionary<string, string>(10);

//Initialiser le dictionnaire
lesPilotes.Add("L. Hamilton", " McLaren Mercedes");
lesPilotes.Add("F. Alonso", " McLaren Mercedes");
lesPilotes.Add("F. Massa", "Ferrari");
lesPilotes.Add("K. Räikkönen", "Ferrari");
lesPilotes.Add("N. Heidfeld", "BMW Sauber");
lesPilotes.Add("R. Kubica", "BMW Sauber");
lesPilotes.Add("G. Fisichella", "Renault");
lesPilotes.Add("H. Kovalainen", "Renault");
lesPilotes.Add("A. Wurz", "Williams");
lesPilotes.Add("M. Webber", "Red Bull");

//lesPilotes.

Afficher(lesPilotes);
// 1 - Pilote : L. Hamilton , Ecurie :  McLaren
Mercedes
// 2 - Pilote : F. Alonso , Ecurie :  McLaren Mercedes
// 3 - Pilote : F. Massa , Ecurie : Ferrari
// 4 - Pilote : K. Räikkönen , Ecurie : Ferrari
// 5 - Pilote : N. Heidfeld , Ecurie : BMW Sauber
// 6 - Pilote : R. Kubica , Ecurie : BMW Sauber
// 7 - Pilote : G. Fisichella , Ecurie : Renault
// 8 - Pilote : H. Kovalainen , Ecurie : Renault
// 9 - Pilote : A. Wurz , Ecurie : Williams
// 10 - Pilote : M. Webber , Ecurie : Red Bull

//Exception "ArgumentNullException" levée lors de
l'ajout d'une clé nulle
try{
   lesPilotes.Add(null, "Ferrari");
}
catch (ArgumentNullException ex){
   Console.WriteLine(System.Environment.NewLine +
ex.Message);
```

```
   //Value cannot be null.
   //Parameter name: key
}

//l'ajout d'une valeur null autorisé
lesPilotes.Add("M. Fekih", null);

//Exception "ArgumentException" levée lors de l'ajout
d'une clé existante
try{
   lesPilotes.Add("L. Hamilton", " McLaren Mercedes");
}
catch (ArgumentException ex){

   Console.WriteLine(System.Environment.NewLine +
ex.Message + System.Environment.NewLine);
   //An item with the same key has already been added.
}

//Supprimer des entrées
lesPilotes.Remove("K. Räikkönen");
lesPilotes.Remove("G. Fisichella");

//Afficher des pilotes
Afficher(lesPilotes);

// 1 - Pilote : L. Hamilton , Ecurie :  McLaren
Mercedes
// 2 - Pilote : F. Alonso , Ecurie :  McLaren Mercedes
// 3 - Pilote : F. Massa , Ecurie : Ferrari
// 4 - Pilote : N. Heidfeld , Ecurie : BMW Sauber
// 5 - Pilote : R. Kubica , Ecurie : BMW Sauber
// 6 - Pilote : H. Kovalainen , Ecurie : Renault
// 7 - Pilote : A. Wurz , Ecurie : Williams
// 8 - Pilote : M. Webber , Ecurie : Red Bull
// 9 - Pilote : M. Fekih , Ecurie :
```

Dans l'exemple précédent, des strings ont été utilisé comme clé. Le Framework reconnait deux strings identiques grâce à l'implémentation de GetHashCode. Dans ce qui suit nous allons voir comment utiliser une classe personnalisée (custom class) comme clé.

Tout d'abord, il faut comprendre comment le dictionnaire du Framework .Net fonctionne pour reconnaitre deux clés identiques. Le principe reste le même, le dictionnaire va calculer la valeur de hachage de la classe et va essayer de retrouver la même valeur dans les clés déjà utilisées dans la table de hachage. Si le dictionnaire ne trouve aucune clé alors il va pourvoir insérer la valeur sans se soucier de la redondance. Dans l'autre cas où le dictionnaire retrouve la même valeur de hachage. Il va vérifier si les deux instances de la classe qui représente la clé sont égaux. Pour cela, il va donc utiliser la méthode Equals.

On va donc dans le code suivant changer la définition de notre classe Pilote en implémentant la classe générique IEquatable et en surchargeant la méthode GetHashCode.

L'interface IEquatable

```
public class Pilote{
    private string nom;

    public string Nom{
        get { return nom; }
        set { nom = value; }
    }

    private string prenom;

    public string Prenom{
        get { return prenom; }
        set { prenom = value; }
    }

    public Pilote(string prenom, string nom){
        this.Nom = nom;
        this.Prenom = prenom;
    }

    public override string ToString(){
        return string.Concat(this.Prenom, " ",
this.Nom);
    }
}

public class PiloteEquatable : Pilote,
IEquatable<PiloteEquatable>{
    public PiloteEquatable(string prenom, string nom)
        : base(prenom, nom){

    }

    public override int GetHashCode(){
        Trace.WriteLine(" GetHashCode appelé ");
        return string.Concat(this.Prenom, " ",
this.Nom).GetHashCode();
    }

    public bool Equals(PiloteEquatable other){
        Trace.WriteLine(" Equals Appelé");
        return this.Nom.Equals(other.Nom) &&
this.Prenom.Equals(other.Prenom);
    }

    public override bool Equals(object obj){
        return Equals((PiloteEquatable)obj);
    }
}

public static class DictionaryDemo{
    public static void ExecuteIEquatable(){
        Dictionary<Pilote, string> lesPilotes = new
Dictionary<Pilote, string>(3);

        //Essayer d'inserer deux pilotes identiques
sans l'implementation de IEquatable
        lesPilotes.Add(new Pilote("Lewis", "Hamilton"),
" McLaren Mercedes");
        lesPilotes.Add(new Pilote("Fernando",
"Alonso"), " McLaren Mercedes");
        lesPilotes.Add(new Pilote("Lewis", "Hamilton"),
" McLaren Mercedes");

        //1 - Pilote : Lewis Hamilton , Ecurie :
McLaren Mercedes
        //2 - Pilote : Fernando Alonso , Ecurie :
McLaren Mercedes
        //3 - Pilote : Lewis Hamilton , Ecurie :
McLaren Mercedes

        //ça fonctionne ; le dictionnaire ne detecte
pas le même pilote
        Afficher(lesPilotes);

        Dictionary<PiloteEquatable, string>
lesPilotesEquatable = new Dictionary<PiloteEquatable,
string>(5);

        lesPilotesEquatable.Add(new
PiloteEquatable("Lewis", "Hamilton"), " McLaren
Mercedes");
        lesPilotesEquatable.Add(new
PiloteEquatable("Fernando", "Alonso"), " McLaren
```

```
Mercedes");

        try{
            lesPilotesEquatable.Add(new
PiloteEquatable("Lewis", "Hamilton"), " McLaren
Mercedes");
        } catch (ArgumentException ex){
            Console.WriteLine(System.Environment.NewLin
e + ex.Message);

            //Ajouter une nouvelle fois Lewis Hamilton
:
            //GetHashCode appelé
               //Equals Appelé

            //Le dictionnaire detecte deux hashcode
identitiques => GetHashCode appelé
            //Il va donc comparer les instaces grâce à
Equals => Equals appelé

            //An item with the same key has already
been added.
        }

        Console.WriteLine(lesPilotesEquatable.ContainsK
ey(new PiloteEquatable("Fernando", "Alonso"))); // True

        lesPilotesEquatable.Remove(new
PiloteEquatable("Fernando", "Alonso"));
    }
}
```

4.3. SortedDictionary

La SortedDictionary est une implémentation particulière de la classe Dictionary précédente. Elle a la particularité d'ordonner les clés de sa table selon une logique définie par l'utilisateur. Cette prise en charge de l'ordonnancement influe sur les performances de la collection.

Quand faut-il choisir la classe SortedDictionary ?

1. Collection typée qui stocke des listes de clé/valeur ordonnées par les clés ; la classe représentant la clé doit implémenter l'interface IComparable. Dans le cas échéant, l'utilisateur peut passer une classe qui implémente IComparer lors de la construction de la collection.
2. N'autorise pas les clés doublons.
3. Plus lente que la Dictionary : Insertion et suppression en O(log(n))

Pour la détection des clés similaires, la SortedDictionary se base sur la méthode CompareTo, membre de l'interface IComparable. Cette méthode est appelée systématiquement lors de l'ajout, suppression et recherche de clés, d'où la perte de performance constatée par rapport à la classe Dictionary.

Par exemple, si on utilise un type primitif du Framework tel que le string, la collection va automatiquement ordonner les éléments selon l'ordre alphabétique.

Manipulation simple d'une SortedDictionary

```
SortedDictionary<string,string> lesPilotes=new
SortedDictionary<string,string>();

lesPilotes.Add("L. Hamilton", "McLaren Mercedes");
lesPilotes.Add("F. Alonso", "McLaren Mercedes");
lesPilotes.Add("F. Massa", "Ferrari");
lesPilotes.Add("K. Räikkönen", "Ferrari");
lesPilotes.Add("N. Heidfeld", "BMW Sauber");
lesPilotes.Add("R. Kubica", "BMW Sauber");
lesPilotes.Add("G. Fisichella", "Renault");
lesPilotes.Add("H. Kovalainen", "Renault");
lesPilotes.Add("A. Wurz", "Williams");
lesPilotes.Add("M. Webber", "Red Bull");

Afficher(lesPilotes);
 //1 - Pilote : A. Wurz , Ecurie : Williams
 //2 - Pilote : F. Alonso , Ecurie : McLaren Mercedes
 //3 - Pilote : F. Massa , Ecurie : Ferrari
 //4 - Pilote : G. Fisichella , Ecurie : Renault
 //5 - Pilote : H. Kovalainen , Ecurie : Renault
 //6 - Pilote : K. Räikkönen , Ecurie : Ferrari
 //7 - Pilote : L. Hamilton , Ecurie : McLaren Mercedes
 //8 - Pilote : M. Webber , Ecurie : Red Bull
 //9 - Pilote : N. Heidfeld , Ecurie : BMW Sauber
 //10 - Pilote : R. Kubica , Ecurie : BMW Sauber
}

public static void ExecuteIEquatable(){
SortedDictionary<PiloteComparable, string> lesPilotes =
new SortedDictionary<PiloteComparable, string>();

//Dans le cas d'une classe personnalisée, il faut
impérativement implementer IComparable
lesPilotes.Add(new PiloteComparable("Lewis",
"Hamilton"), "McLaren Mercedes");

//la duplication est detectée par CompareTo et non pas
par Equals et GetHashCode
try{
    lesPilotes.Add(new PiloteComparable("Lewis",
"Hamilton"), "McLaren Mercedes");
}
catch (ArgumentException ex){

    Console.WriteLine(System.Environment.NewLine +
ex.Message + System.Environment.NewLine);
    //An entry with the same key already exists.
}

lesPilotes.Add(new PiloteComparable("Fernando",
"Alonso"), "McLaren Mercedes");
lesPilotes.Add(new PiloteComparable("Felipe", "Massa"),
"Ferrari");
lesPilotes.Add(new PiloteComparable("Kimi",
"Räikkönen"), "Ferrari");
lesPilotes.Add(new PiloteComparable("Nick",
"Heidfeld"), "BMW Sauber");
lesPilotes.Add(new PiloteComparable("Robert",
"Kubica"), "BMW Sauber");
lesPilotes.Add(new PiloteComparable("Giancarlo",
"Fisichella"), "Renault");
lesPilotes.Add(new PiloteComparable("Heikki",
"Kovalainen"), "Renault");
lesPilotes.Add(new PiloteComparable("Alexander",
"Wurz"), "Williams");
lesPilotes.Add(new PiloteComparable("Mark", "Webber"),
"Red Bull");

Afficher(lesPilotes);
//1 - Pilote : Alexander Wurz , Ecurie : Williams
//2 - Pilote : Felipe Massa , Ecurie : Ferrari
//3 - Pilote : Fernando Alonso , Ecurie : McLaren
Mercedes
//4 - Pilote : Giancarlo Fisichella , Ecurie : Renault
//5 - Pilote : Heikki Kovalainen , Ecurie : Renault
//6 - Pilote : Kimi Räikkönen , Ecurie : Ferrari
//7 - Pilote : Lewis Hamilton , Ecurie : McLaren
Mercedes
//8 - Pilote : Mark Webber , Ecurie : Red Bull
//9 - Pilote : Nick Heidfeld , Ecurie : BMW Sauber
//10 - Pilote : Robert Kubica , Ecurie : BMW Sauber
```

```
//ContainsKey utilise que CompareTo
Console.WriteLine(lesPilotes.ContainsKey(new
PiloteComparable("Robert", "Kubica")));
//True

//Remove utilise que CompareTo
lesPilotes.Remove(new PiloteComparable("Giancarlo",
"Fisichella"));

Console.WriteLine(Environment.NewLine + lesPilotes[new
PiloteComparable("Mark", "Webber")]);
//Red Bull
```

L'avantage avec l'exemple précédent est que le Framework sait comment comparer deux string et donc situer une chaîne de caractères par rapport à une autre. Ce n'est pas le cas avec une classe personnalisée. Pour remédier à ce problème et "apprendre" au Framework comment comparer deux instances d'une classe personnalisée, il faut dans ce cas implémenter l'interface générique IComparable et par la suite redéfinir la méthode CompareTo.

Voici une implémentation de IComparable de notre classe de test Pilote :

L'interface générique IComparable

```
public class PiloteComparable : Pilote,
IComparable<PiloteComparable>{
    public PiloteComparable(string prenom, string nom)
        : base(prenom, nom){

    }

    public int CompareTo(PiloteComparable other){
        Trace.WriteLine("CompareTo appelé");

        if (this.Prenom.CompareTo(other.Prenom) == 0)
            return this.Nom.CompareTo(other.Nom);
        else return
this.Prenom.CompareTo(other.Prenom);
    }
}

Maintenant, on va utiliser la classe PiloteComparable
comme clé de notre collection SortedDictionary :
Manipulation d'une SortedDictionary

SortedDictionary<PiloteComparable, string> lesPilotes =
new SortedDictionary<PiloteComparable, string>();

//Dans le cas d'une classe personnalisée, il faut
impérativement implementer IComparable
lesPilotes.Add(new PiloteComparable("Lewis",
"Hamilton"), "McLaren Mercedes");

//la duplication est detectée par CompareTo et non pas
par Equals et GetHashCode
try{
    lesPilotes.Add(new PiloteComparable("Lewis",
"Hamilton"), "McLaren Mercedes");
} catch (ArgumentException ex){
    Console.WriteLine(System.Environment.NewLine +
ex.Message + System.Environment.NewLine);
    //An entry with the same key already exists.
}

lesPilotes.Add(new PiloteComparable("Fernando",
"Alonso"), "McLaren Mercedes");
lesPilotes.Add(new PiloteComparable("Felipe", "Massa"),
"Ferrari");
lesPilotes.Add(new PiloteComparable("Kimi",
"Räikkönen"), "Ferrari");
lesPilotes.Add(new PiloteComparable("Nick",
"Heidfeld"), "BMW Sauber");
lesPilotes.Add(new PiloteComparable("Robert",
"Kubica"), "BMW Sauber");
lesPilotes.Add(new PiloteComparable("Giancarlo",
"Fisichella"), "Renault");
lesPilotes.Add(new PiloteComparable("Heikki",
"Kovalainen"), "Renault");
lesPilotes.Add(new PiloteComparable("Alexander",
"Wurz"), "Williams");
lesPilotes.Add(new PiloteComparable("Mark", "Webber"),
"Red Bull");

Afficher(lesPilotes);
//1 - Pilote : Alexander Wurz , Ecurie : Williams
//2 - Pilote : Felipe Massa , Ecurie : Ferrari
//3 - Pilote : Fernando Alonso , Ecurie : McLaren
Mercedes
//4 - Pilote : Giancarlo Fisichella , Ecurie : Renault
//5 - Pilote : Heikki Kovalainen , Ecurie : Renault
//6 - Pilote : Kimi Räikkönen , Ecurie : Ferrari
//7 - Pilote : Lewis Hamilton , Ecurie : McLaren
Mercedes
//8 - Pilote : Mark Webber , Ecurie : Red Bull
//9 - Pilote : Nick Heidfeld , Ecurie : BMW Sauber
//10 - Pilote : Robert Kubica , Ecurie : BMW Sauber

//ContainsKey utilise que CompareTo
Console.WriteLine(lesPilotes.ContainsKey(new
PiloteComparable("Robert", "Kubica")));
//True

//Remove utilise que CompareTo
lesPilotes.Remove(new PiloteComparable("Giancarlo",
"Fisichella"));

Console.WriteLine(Environment.NewLine + lesPilotes[new
PiloteComparable("Mark", "Webber")]);
//Red Bull
```

4.4. OrderedDictionary

La OrderedDictionary est une implémentation particulière de la table de hachage. Elle permet de disposer d'un accès indexé en consultation et suppression. Elle est généralement utile pour contrôler l'ordre des éléments dans le dictionnaire mais reste néanmoins une collection non typée.

Quand faut-il choisir la classe OrderedDictionary ?

1. Collection non typée qui stocke des listes de clé/valeur. L'accès aux couples est possible soit par clé ou par index
2. N'autorise pas les clés doublons.
3. Rapide : accès et suppression en O(1)

Le code qui suit montre comment utiliser les fonctions d'insertion et de suppression par index et par clé dans la classe OrderedDictionary :

Manipulation d'un OrderedDictionary

```
OrderedDictionary lesPilotes = new OrderedDictionary();

lesPilotes.Add("L. Hamilton", "McLaren Mercedes");
lesPilotes.Add("F. Alonso", "McLaren Mercedes");
lesPilotes.Add("F. Massa", "Ferrari");

//insertion indexée
lesPilotes.Insert(3, "K. Räikkönen", "Ferrari");
lesPilotes.Insert(4, "N. Heidfeld", "BMW Sauber");
lesPilotes.Insert(5, "R. Kubica", "BMW Sauber");

lesPilotes.Add("G. Fisichella", "Renault");
```

```
lesPilotes.Add("H. Kovalainen", "Renault");
lesPilotes.Add("A. Wurz", "Williams");
lesPilotes.Add("M. Webber", "Red Bull");

Afficher(lesPilotes);
//1 - Pilote : L. Hamilton , Ecurie : McLaren Mercedes
//2 - Pilote : F. Alonso , Ecurie : McLaren Mercedes
//3 - Pilote : F. Massa , Ecurie : Ferrari
//4 - Pilote : K. Räikkönen , Ecurie : Ferrari
//5 - Pilote : N. Heidfeld , Ecurie : BMW Sauber
//6 - Pilote : R. Kubica , Ecurie : BMW Sauber
//7 - Pilote : G. Fisichella , Ecurie : Renault
//8 - Pilote : H. Kovalainen , Ecurie : Renault
//9 - Pilote : A. Wurz , Ecurie : Williams
//10 - Pilote : M. Webber , Ecurie : Red Bull

//Ajouter une nouvelle entrée à la tête du dictionnaire
lesPilotes.Insert(0, "Lewis Hamilton", "McLaren
Mercedes");
```

```
//Modification indexée
lesPilotes[0] = "McLaren";
//Modification par clé
lesPilotes["Lewis Hamilton"] = "McLaren Mercedes";

Afficher(lesPilotes);
//1 - Pilote : Lewis Hamilton , Ecurie : McLaren
Mercedes
//2 - Pilote : L. Hamilton , Ecurie : McLaren Mercedes
//3 - Pilote : F. Alonso , Ecurie : McLaren Mercedes
//4 - Pilote : F. Massa , Ecurie : Ferrari
//5 - Pilote : K. Räikkönen , Ecurie : Ferrari
//6 - Pilote : N. Heidfeld , Ecurie : BMW Sauber
//7 - Pilote : R. Kubica , Ecurie : BMW Sauber
//8 - Pilote : G. Fisichella , Ecurie : Renault
//9 - Pilote : H. Kovalainen , Ecurie : Renault
//10 - Pilote : A. Wurz , Ecurie : Williams
//11 - Pilote : M. Webber , Ecurie : Red Bull
```

Retrouvez la suite de l'article de Mehdi Fekih en ligne : Lien43

Les livres .NET

Asp.net 2.0 Web Site Programming: Problem-design-solution

* The unique Problem-Design-Solution approach shows Web developers how to integrate all the new functionality of ASP.NET 2.0 into a single real-world project that reflects problems and solutions they face daily
* Walks readers through the development of a complete ASP.NET 2.0 Web site that boasts most of the features a user can expect to find in an e-commerce site: photo galleries; opinion polls, forums, and newsletters; an e-commerce section with shopping cart; and order management functions
* The implementation of each feature demonstrates new functions introduced by ASP.NET 2.0, such as master pages, Web parts, theming, membership, personalization, and much more

Critique du livre par Philippe Vialatte

Lorsqu'il m'a fallu passer des winforms 1.1 à asp.net 2.0, je me suis trouvé confronté à un dilemme. Soit, d'un coté, prendre un livre de référence complet sur ASP.Net, qui aurait présenté le Framework de fond en comble, mais sans forcément le rattacher à des exemples, soit prendre un livre plus orienté "débutant", qui aborde les problèmes sans aller au fond, mais d'une façon plus didactique.

Le but de ce livre est de construire un site web Asp.Net complet, depuis le concept jusqu'à une implémentation complète, avec système de paiement en ligne, newsletter, interface d'administration, etc...

L'approche de chaque chapitre est la suivante : le problème est posé, puis étudié en détail, les différents choix d'implémentation expliqués, argumentés (avec les forces et les faiblesses de chaque implémentations), puis mis en place.

Attention par contre, ce livre présuppose une connaissance de base du Framework et du développement web. Ce n'est pas le premier livre à lire si vous n'avez jamais développé.

Retrouvez ce livre sur la rubrique .NET : Lien44

Les blogs .NET

Publication en grande partie du code source du Framework .NET avec la sortie de Visual Studio 2008

C'est grâce à un article de Scott Guthrie que nous prennons connaissance des intentions de Microsoft de rendre une grande partie du code source du Framework .NET accessiblr.

Nous pourrons donc librement regarder le fonctionnement des librairies de classes .NET de bases (System, System.IO, System.Collections, System.Configuration, System.Threading, System.Net, System.Security, System.Runtime, System.Text, etc), ASP.NET (System.Web), Windows Forms (System.Windows.Forms), ADO.NET (System.Data), XML (System.Xml), and WPF (System.Windows).

Un bon point pour les développeurs qui pourront mieux comprendre comment le Framework à été créé. Mais rendre le framework .NET Open Source va permettre un debuguage plus en profondeur, directement dans le framework lui même.

Pour plus d'information je vous propose l'article de l'origine de cette annonce : Lien45

Retrouvez ce billet sur le blog de la rédaction .NET : Lien46

Les derniers tutoriels et articles

Découverte du débogueur DDD

Ce tutoriel vous permet d'aborder le debogueur DDD et suppose que vous savez déjà utiliser un debogueur. Cela sous-entend que vous savez ce qu'est une pile d'appels, un point d'arrêt, ...

1. Introduction

Un debogueur est un outil qui permet d'exécuter un programme en le contrôlant presque intégralement. L'objectif d'un tel programme est de permettre au développeur de rechercher les erreurs de programmation qu'il a commise et qui ne sont pas détectées par le compilateur.

Le debogueur DDD est en fait une surcouche graphique des debogueurs en mode texte tel "gdb". Ce tutorial se base sur une approche C & C++ du debogueur, mais il peut etre utilisé pour d'autres langages.Pour l'installation de cet outil, je vous laisse le soin de vous fier au mode d'installation de votre distribution favorite.

2. Préparer un projet pour le débogage

2.1. Pour déboguer, il faut compiler en... debug

Pour commencer, il faut utiliser le compilateur gcc/g++ suivant que vous conceviez une application C ou bien C++. A la ligne de compilation, il faut rajouter l'option "-g" comme par exemple :

```
g++ -o test test.cpp -g
```

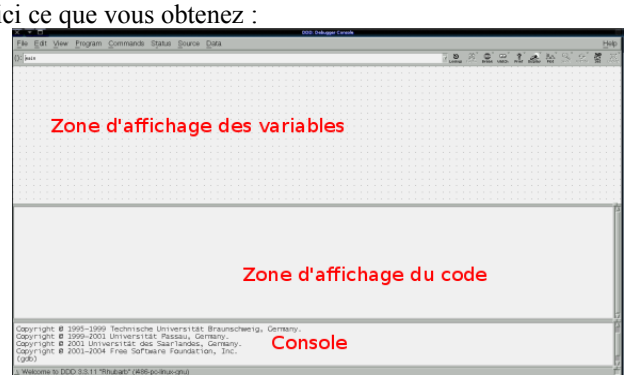
Ce mode de compilation correspond au mode "debug" pour les utilisateurs de visual studio.

2.2. Lancement de DDD

Maintenant que la compilation, est effectuée il faut lancer DDD, en y spécifiant ou non le chemin de l'exécutable à deboguer. Pour cela il suffit de saisir la commande suivante:

```
ddd [chemin_executable]
```

Il est possible d'y associer plusieurs options qui permettent de choisir le debogueur associé. Pour plus d'informations je vous laisse consulter le manpage associé (Lien47).
Voici ce que vous obtenez :



2.3. Ouverture de la partie à déboguer

Il est possible de deboguer plusieurs choses : un exécutable, un dump, ... Pour ouvrir ce que vous souhaitez, rendez vous dans le menu fichier (si vous n'avez rien spécifié dans la ligne de commande pour lancer ddd).

Dans notre cas, nous allons ouvrir un exécutable de base obtenu via le code source suivant :

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

using namespace std;

void printtest(string strCmd){
	cout << strCmd << endl;
}

int main(int argc, char **pArgs){
	if (argc > 1){
		string strCmd = "le premier paramètre
est : ";
		strCmd += pArgs[1];
		printtest(strCmd);
	}
	else{
		printtest("commande vide !");
	}

	return EXIT_SUCCESS;
}
```

2.4. Passer des arguments à la ligne de commande

Maintenant que DDD est lancé, il faut indiquer la ligne de commande qui va être passée à l'application en cours de débogage.

Attention, les arguments ne sont spécifiés qu'au lancement de l'application, il faut donc indiquer les points d'arrêts nécessaires avant.

Pour saisir la ligne d'arguments, rendez-vous dans le menu "Program" puis l'item "Run..." (ou plus simplement la touche F2).

Il vous suffit de saisir la liste d'arguments dans la partie "Run with Arguments" puis de lancer l'exécution du programme en cliquant sur le bouton "run". Vous devriez voir dans la partie console soit votre argument, soit le message "commande vide!".

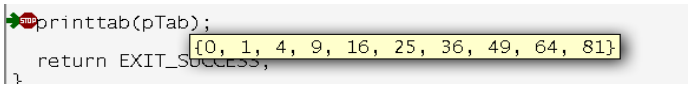
3. Les outils de débogage

3.1. Les espions

Une première fonctionnalité primordiale du débogueur est la possibilité de scruter la valeur des variables, et ce à tout moment. Il existe plusieurs façons de visualiser ces valeurs, lorsque le programme est stoppé en mode débogage.

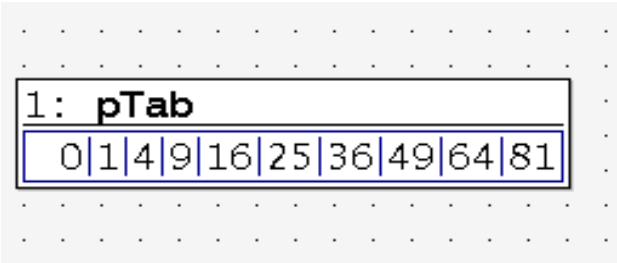
3.1.1. Les infobulles

La première manière de visualiser une variable est de simplement placer le pointeur de la souris sur celle-ci : une infobulle vous indiquera sa valeur. S'il s'agit d'un tableau ou d'une chaîne de caractères, vous pouvez même voir le contenu d'un tableau ou de la chaîne :

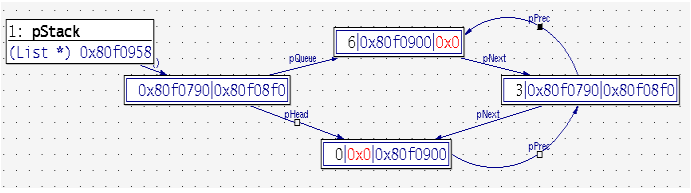


3.1.2. La zone d'affichage des variables

Il est possible d'afficher les variables de manière permanente pour pouvoir visualiser leurs évolutions. Pour cela, il faut effectuer un clic droit sur la variable désirée et sélectionner l'option "display". L'option "print" affiche juste dans la zone "console" la valeur de la variable. Les options "display *" et "print *" permettent, comme vous vous en doutez, non pas d'afficher la valeur de la variable mais la valeur contenu à l'adresse que contient la variable. Pour être plus clair, si vous prenez une variable représentant un tableau "display" montrera le contenu du tableau et "display *" le premier élément. Lorsque l'option "display" a été invoquée, vous verrez apparaître votre variable dans la zone de visualisation des variables :



Un cas spécifique ne peut être affiché qu'en utilisant cette zone : il s'agit de l'affichage d'une structure "récursive" (pile, liste, ...) En effet, comme il s'agit de pointeurs, il faut afficher la variable puis dérouler manuellement les pointeurs comme suit :



Pour dérouler manuellement, il suffit de double cliquer sur le membre de la structure que l'on souhaite afficher. Dans le cas présent voici la déclaration de la structure liste utilisée :

```
struct stElement{
  int nValue;
  struct stElement *pNext;
  struct stElement *pPrec;
};

struct stList{
 struct stElement *pHead;
 struct stElement *pQueue;
};

typedef struct stElement Element;
typedef struct stList List;
```

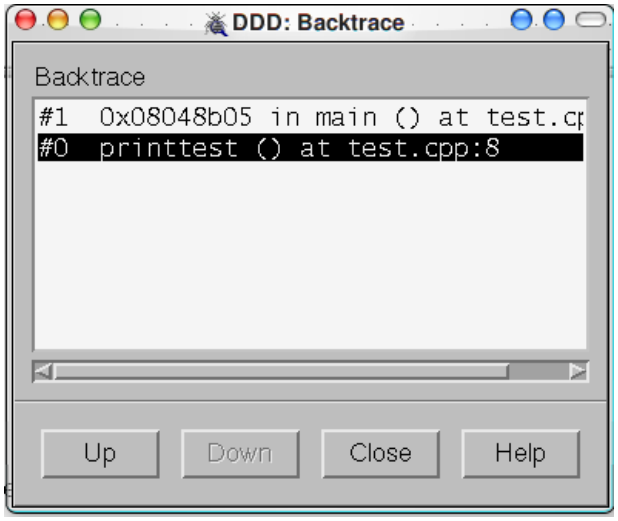
On retrouve donc les 3 membres par élément de la liste :

- un entier représentant la valeur
- un hexadécimal représentant l'adresse de l'élément suivant
- un hexadécimal représentant l'adresse de l'élément précédant

Comme vous pouvez vous en douter, l'adresse 0x0 correspond à la valeur NULL.

3.2. La pile des appels

La pile des appels peut être obtenue grâce au menu "Status" puis à l'entrée "Backtrace". Vous arrivez ainsi sur une fenêtre de ce type :



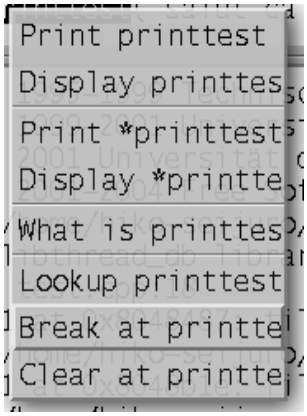
En double cliquant sur une entrée, DDD vous place automatiquement sur la ligne pointée par l'entrée dans la pile : la fonction appelante en général.

3.3. Les points d'arrêt

On distingue 2 types de points d'arrêt : ceux qui sont à l'entrée d'une fonction et ceux qui sont placés à des points précis du code. En fait les premiers ne sont qu'un cas particulier des seconds.

3.3.1. Points d'arrêt sur une entrée de fonction

Pour ajouter un point d'arrêt à l'entrée d'une fonction, on peut se placer dans le contexte d'une fonction appelante, sélectionner le nom de la fonction (un double clic sur ce nom suffit !) et effectuer un clic droit pour obtenir le menu suivant :



Il suffit alors de sélectionner la ligne "Break at ...", et le point d'arrêt sera placée sur la première ligne codante de la fonction comme par exemple :

```
void printtest(string strCmd)
{
  cout << strCmd << endl;
}
```

Il faut être conscient qu'à chaque appel de la fonction, un arrêt sera effectué et pas seulement celui effectué par la fonction appelante

3.3.2. Points d'arrêt sur une ligne précise du code

Pour assigner un point d'arrêt à une ligne de code précise, il faut effectuer un clic droit et choisir l'élément "Set Breakpoint". Il est possible d'appliquer un point d'arrêt temporaire (il se détruit dès qu'il atteint) avec la commande "Set Temporary Breakpoint" :

```
else
        {
          printtest("sa
isdufosiduf oisudfho ze
        }
```

Comme vous pouvez le constatez, la "petite croix" différencie un point d'arrêt temporaire d'un point d'arrêt classique.

3.3.3. Se positionner sur une fonction précise

Pour se positionner dans une fonction précise, il est possible de faire une recherche de cette fonction. Pour cela, il faut se servir de la barre située en dessous de la barre de menu :

Lors du clic sur "Lookup" ou de la validation (en appuyant sur la touche "Entrée"), DDD place la première ligne de la fonction au milieu de la zone d'affichage du code. Ceci est très utile si on souhaite mettre un point d'arrêt dans une fonction n'étant pas appelée directement (ou de manière rapidement visible)

3.4. L'exécution pas à pas

DDD possède une petite zone de commande qui permet de contrôler l'exécution du programme :

Voici la correspondance de chaque élément du panneau de contrôle :

- **Run** : permet de lancer l'exécution du programme.
- **Interrupt** : permet de stopper l'exécution du programme. Il ne s'agit pas de mettre le programme en pause mais de l'arrêter complètement
- **Step** : permet d'effectuer un pas en avant dans l'exécution y compris en entrant dans le corps d'une fonction (méthode) appelée
- **Stepi** : permet d'effectuer un pas en avant dans l'exécution d'une instruction exactement
- **Next** : permet de continuer l'exécution sans rentrer dans le corps des fonctions appelées, le pas étant la ligne de code
- **Nexti** : permet de continuer l'exécution sans rentrer dans le corps des fonctions appelées, le pas étant l'opcode assembleur
- **Until** : permet de laisser une boucle s'exécuter et continuer le deboggage à partir de l'instruction qui suit.
- **Finish** : permet de continuer l'exécution jusqu'à rencontrer un "return"
- **Cont** : permet de continuer l'exécution jusqu'à rencontrer un nouveau point d'arrêt
- **Kill** : permet de stopper le programme de façon très brutal !
- **Up** : permet de remonter dans la pile d'appel
- **Down** : permet de descendre dans la pile d'appel
- **Undo** : permet d'annuler l'action effectuée
- **Redo** : permet de refaire l'action qui vient d'être annulée
- **Edit** : permet d'éditer le programme source
- **Make** : permet d'appeler le programme "make"

L'exécution "Pas à Pas" permet de scruter très précisement un programme. Cela a pour objectif, en général, de surveiller le comportement d'une variable localement.

3.5. Modifier & relancer l'exécution

Il s'agit d'un des points faibles de DDD : il est impossible de modifier un programme et de reprendre directement où l'exécution s'est arrêté. En revanche, il est possible de modifier un programme, de le recompiler et de relancer avec les mêmes arguments. La procédure est très simple, il suffit d'éditer le source (avec la zone de commande "Edit" ou de manière extérieure), de le compiler (avec le bouton "Make" de la zone de commande, ou un appel manuel) et enfin utiliser le raccourci F2 et sélectionner les arguments.

4. Configuration plus poussée de DDD

Le but de cette section est de présenter comment adapter DDD à vos besoins. Comme pour la plupart des logiciels sous linux, ces configurations seront sauvés dans le répertoire ~/.ddd. Si vous devez effectuer un formatage ou une autre opération supprimant vos configurations n'oubliez pas de copier ce répertoire si vous souhaitez conserver les modifications apportées à DDD.

4.1. Rendre la sauvegarde de la configuration effective

Il faut effectuer une petite manipulation pour que la sauvegarde soit effective : il faut cocher l'option du menu Edit Save Options.

4.2. Configurer DDD

Pour configurer DDD, rendez-vous dans le menu **Edit->Preferences**.

Chaque onglet correspond à un point précis de configuration :

- **General** permet de configurer l'interface de DDD
- **Source** permet de configurer la zone de commande ainsi que la zone du code source
- **Data** permet de configurer l'affichage des variables dans

la zone dédiée
- **Startup** permet de configurer la mise en place de DDD lors de son démarrage
- **Fonts** permet de configurer les polices de caractères utilisées
- **Helpers** permet de configurer les programmes extérieurs qui peuvent être appelés

Les "importants" sont Data et Helpers. Nous allons donc expliquer ces 2 entités. Pour plus de précision sur les autres, je vous conseille d'aller lire le manuel de DDD dont l'adresse figure à la fin de cet article.

4.2.1. Data

Les éléments les plus importants sont :
- **Show** permet de choisir les éléments à afficher surtout concernant les structures arborescentes : les liens (edge hints), le nom des liens (edge annotations)
- **Placement** permet de définir comment les variables seront disposées par défaut (de haut en bas, de gauche à droite, ...)
- **Display Two-dimensional Arrays as Table** permet, comme son nom l'indique, d'afficher les tableaux à 2 dimensions sous la forme d'une table. Attention cela ne marche pas pour les tableaux alloués dynamiquement !

Le bouton **Themes** permet de pousser la configuration de l'affichage des variables plus loin, je vous laisse le soin d'y consacrer une recherche sur le manuel si vous souhaitez y consacrer un peu de temps.

4.2.2. Helpers

La partie "Helpers" permet de configurer les programmes appelés par DDD.

Les programmes extérieurs les plus utilisés sont :
- **Edit source** qui permet de choisir le programme qui va être appelé lors du clic sur le bouton "Edit" de la zone de commande. La syntaxe a respecter est la suivante : <nomediteur> @FILE@
- **Plot** permet de définir la ligne de commande que va utiliser gnuplot. Je conseille de laisser cette ligne de commande telle quelle mais le manuel pourra vous aider si souhaitez manipuler de manière plus poussée gnuplot.
- **Plot Window** permet de définir si gnuplot est lancé en interne ou en externe. Je conseille de le mettre en externe car DDD peut planter facilement si il est lancé en interne.

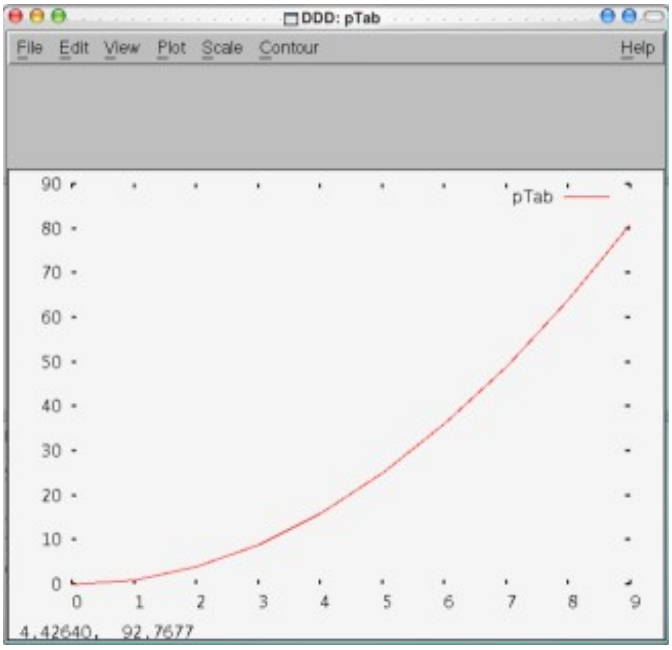
5. Les possibilités avancées

5.1. Visualisation des tableaux sous formes de courbes

Pour cette partie, il faut installer un outil complémentaire qui s'appelle gnuplot. Nous allons utiliser le programme exemple suivant.

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

void printtab(const int *pTab){
  int i;
    for(i=0;i<10;i++){
          printf("tab[%d] = %d\n", i, pTab[i]);
        }
}

int main(int argc, char **pArgs){
  int pTab[10];
  int i;

  for(i=0;i<10;i++){
          pTab[i] = i*i;
        }

  printtab(pTab);

  return EXIT_SUCCESS;
}
```

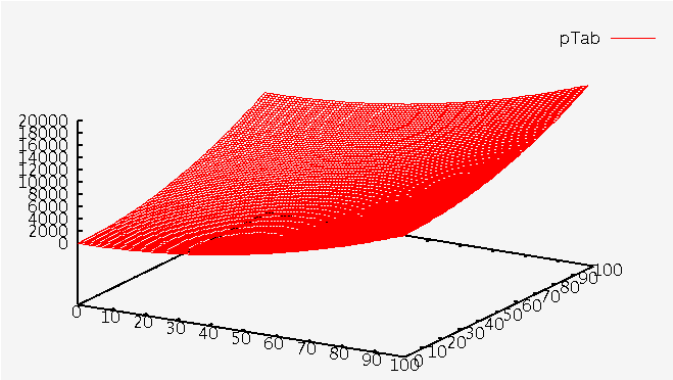
Si on place un point d'arrêt sur la fonction "printtab", qu'on sélectionne la variable pTab (comme si on sélectionnait un texte) puis on choisit l'élément "plot" dans la barre d'outil. On obtient alors ceci :



Il est aussi possible d'afficher des matrices sous gnuplot :

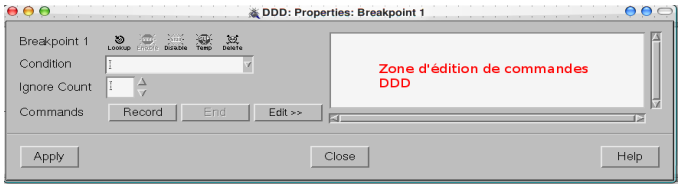
```
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char **pArgs){
  int pTab[100][100];
  int i,j;

  for(i=0;i<100;i++){
      for (j=0;j<100;j++){
          pTab[i][j] = j*j+i*i;
      }
  }

  return EXIT_SUCCESS;
}
```

Avec un point d'arrêt placé sur l'instruction return et lors de la mise en place de la suite de la procédure on obtient :



5.2. Les points d'arrêts conditionnels

Il est possible de positionner des points d'arrêts qui ne seront actifs que sous certaines conditions (variable = une valeur précise, ...). Pour cela, il faut créer un point d'arrêt comme vu précédemment, puis effectuer un clic droit sur le point d'arrêt et sélectionner l'option "properties". Vous obtiendrez la boite de dialogue suivante :



Dans la zone "condition", vous pouvez saisir la condition qui rendra le point d'arrêt actif. Attention il s'agit de condition sous la forme "C" (égalité est "==").

5.3. Associer des commandes à des points d'arrêts

Dans la boite de dialogue des propriétés d'un point d'arrêt, il y a une zone "Command". Cette zone permet de décrire une série de commandes spécifiques à DDD lors de l'arrêt du programme sur le point correspondant. Pour rendre visible cette zone, si elle ne l'est pas, il faut cliquer sur le bouton "edit".

6. Conclusion

En conclusion, on peut dire que DDD n'est peut pas aussi évolué que visual c++ mais possède beaucoup atouts. Certaines fonctionnalités du débogueur de Microsoft ne sont pas présentes mais le développeur, dès qu'il maîtrise bien DDD, peut s'en passer. L'outil est donc un incontournable du monde libre.

De plus, il faut bien se rappeler qu'il s'agit d'un "front end" qui peut s'adapter à d'autres debugger tel jdb,... ce qui, évidemment, se fait en occultant certaines fonctionnalités précises pour chaque débugger.

7. Pour aller plus loin...

Si vous voulez utiliser DDD de manière plus poussée, je vous conseille de vous rendre sur le manuel de DDD (Lien48).

Retrouvez l'article d'Hiko-seijuro en ligne : Lien49

Blogs GTK

GTK+ 2.12

Moins de quatre mois après l'annonce de GTK+ 2.11.0, la version stable est aujourd'hui disponible !

Comme prévu, les GtkTooltip (Lien50) remplacent les GtkTooltip**s**. La différence de nom est subtile, mais l'API a été simplifiée et permet plus de chose, comme utiliser la mise en forme du texte, l'inclusion d'image et l'utilisation direct d'un widget de votre choix !

Seconde nouveauté, la plus grosse et celle que j'attendais avec impatience : GtkBuilder (Lien51). Pour ceux qui ne connaissent pas, il s'agit d'une API qui permet de créer une interface graphique à partir d'un fichier XML. L'avantage c'est que ce fichier XML peut être généré à l'aide d'un logiciel, le plus connu étant Glade (Lien52).

Et enfin quelques widgets supplémentaires :
- GtkScaleButton : Lien53
- GtkVolumeButton : Lien54
- GtkRecentAction : Lien55

Pour connaître l'ensemble des nouveautés : GTK+ 2.12 released (Lien56)

Comme souvent, quelques semaines avant la sortie d'une nouvelle version de GTK+, la glib en fait de même avec, le mois dernier, la sortie de la version 2.14 qui apporte deux nouveautés :
- GSequences (Lien57) : il s'agit d'une structure de données de type liste, mais représenté en interne par un arbre binaire balancé,
- GRegex (Lien58) : qui permet d'utiliser les expressions régulières.

GLib 2.14 released (Lien59)

Au final rien d'exceptionnel mais des petits plus qui devrait grandement simplifier les choses

Retrouvez ce billet sur le blog de Gege2061 GTK : Lien60

Les derniers tutoriels et articles

Utiliser les variables tableaux en VBA Excel

Cet article propose une initiation aux variables tableaux, en VBA Excel.

1. Introduction

Les variables tableaux servent à stocker et manipuler des groupes de données du même type.
Les éléments du tableau sont indexés séquentiellement. Chaque élément est identifiable par un numéro d'indice. Les modifications apportées à une donnée du tableau n'affectent pas les autres éléments.
Créer un tableau de taille X revient en quelque sort à déclarer X variables différentes, en une fois.

Un tableau est constitué d'une ou plusieurs dimensions. Le nombre d'éléments pour chaque dimension est défini par des limites inférieures et supérieures. La taille et les dimensions peuvent être fixes (statiques) ou libres (dynamiques).

Seuls les tableaux dynamiques peuvent modifier leur taille et leur dimension en cours de procédure.

Passer par un tableau n'est pas une obligation. Toutefois, cette méthode permet des gains de temps significatifs, notamment pour la manipulation des grands groupes de données. Sur les grandes collections, il convient d'éviter l'énumération qui est très lente.

Comparez la rapidité d'exécution du code entre:

Vba
```
Dim ObjCell As Range

For Each ObjCell In Range("A1:J65535").Cells
    ObjCell.Value = ObjCell.Value * 2 + 3
Next
```

et

Vba
```
'Exemple issu de la FAQ Excel
'http://excel.developpez.com/faq/?page=Cellule#TabCellulesVariant
'Auteur: Bidou

Dim Montab As Variant, cmpt1 As Long, cmpt2 As Long

Montab = Range("A1:J65535").Value

For cmpt1 = LBound(Montab, 1) To UBound(Montab, 1)
    For cmpt2 = LBound(Montab, 2) To UBound(Montab, 2)
        Montab(cmpt1, cmpt2) = Montab(cmpt1, cmpt2) * 2 + 3
    Next cmpt2
Next cmpt1

Range("A1:J65535").Value = Montab
```

Le deuxième code s'exécute environ 20 fois plus vite.

2. Description

2.1. Les tableaux de taille fixe

Un tableau est dit de taille fixe lorsque ses dimensions sont prédéfinies au moment de la déclaration de la variable.

La ligne de code suivante montre comment déclarer un tableau fixe.

```
Dim NomTableau(2) As String
```

L'indice **2** spécifie la taille du tableau. L'indice inférieur d'un tableau peut commencer à 0 ou à 1 en fonction de la définition de l'instruction **Option Base**.
Pour obtenir plus de détails sur la gestion des indices, consultez les chapitres:

- Option Base
- LBound et UBound

As String définit le type de données.
Les tableaux se déclarent de la même façon que les autres variables. Tous les types de variables peuvent être utilisés afin de déclarer un tableau. Déclarez explicitement vos tableaux, avec un type de données adapté, afin d'optimiser l'utilisation de l'espace mémoire.

Dans le premier exemple ci-dessous, 0 est le plus petit index du tableau, et 2 l'index le plus élevé. Cela signifie que le tableau pourra contenir 3 éléments.
Cette procédure simplifiée montre comment alimenter les éléments du tableau et comment boucler sur ces mêmes éléments afin d'en lire le contenu.

Vba
```
Option Explicit
Option Base 0

Sub MonPremierTableau()
    'Définit la taille du tableau et le type de données.
    Dim NomTableau(2) As String
    Dim i As Integer

    'Alimente les éléments du tableau
    NomTableau(0) = "a"
    NomTableau(1) = "b"
    NomTableau(2) = "c"

    'Boucle sur les éléments du tableau pour lire leur contenu
```

```
    For i = 0 To 2
        MsgBox NomTableau(i)
    Next i
End Sub
```

2.2. Les tableaux dynamiques

Si vous ne spécifiez pas la dimension au moment de la déclaration de la variable, le tableau est appelé dynamique.

Ce type de tableau est utilisé lorsque l'on ne connaît pas à l'avance la taille et/ou les dimensions à attribuer. Celles ci seront précisées en cours de procédure, grâce à l'instruction **ReDIm**.

Vous pouvez ainsi changer aussi souvent que vous le souhaitez:

- Le nombre de dimensions (Voir le chapitre 'Les tableaux multidimensionnels').
- Le nombre d'éléments (Voir les chapitres 'ReDim', 'Option Base').
- Les limites supérieures et inférieures de chaque dimension (Voir les chapitres 'Option Base', 'LBound et UBound').

Pour rendre un tableau dynamique, n'indiquez aucune valeur entre les parenthèses (contrairement aux tableaux fixes) lorsque vous déclarez la variable.

```
Dim NomTableau() As String
```

Cet exemple montre comment définir la taille du tableau à partir de la variable "i".
Vous remarquerez qu'**Option Base** n'est pas précisé: 0 est donc le plus petit indice (index) du tableau.

Vba
```
Option Explicit

Sub MonDeuxiemeTableau()
    'Définit le type de données pour le tableau.
    Dim NomTableau() As String
    Dim i As Integer, j As Integer

    i = 2
    'Définit la taille du tableau
    ReDim NomTableau(i)


    'Alimente les éléments du tableau
    For j = 0 To UBound(NomTableau)
        NomTableau(j) = Chr(65 + j)
    Next j


    'Boucle sur les éléments du tableau
    For j = 0 To UBound(NomTableau)
        MsgBox NomTableau(j)
    Next j
End Sub
```

C'est **ReDim** qui attribue de l'espace mémoire aux tableaux dynamiques.
Consultez le chapitre réservé à cette instruction pour plus de détails.

2.3. Les tableaux multidimensionnels

Tous les exemples vus jusqu'à présent étaient à dimension unique:

```
Dim NomTableau(2) As String 'déclare un tableau fixe
ReDim LeTableau(i) 'Redimensionne un tableau dynamique
```

De la même manière, il est possible de créer des tableaux multidimensionnels statiques et dynamiques.
Vous pouvez déclarer jusqu'à 60 dimensions dans une variable tableau.

Il suffit d'insérer des virgules pour séparer chaque dimension, quand vous déclarez le tableau.
Cet exemple déclare un tableau de 3 dimensions:

```
Dim NomTableau(5, 10, 20) As Integer
```

Le nombre total d'éléments disponible est donc le produit des tailles de toutes les dimensions.

Lorsque vous souhaitez agrandir un tableau dynamique tout en conservant les données existantes, seule la dernière dimension peut être redimensionnée
(Voir le chapitre ReDim Preserve pour plus de détails).

Utilisez des boucles imbriquées pour manipuler les tableaux à plusieurs dimensions.

Vba
```
Option Explicit

Sub ExempleTableau_MultiDimensionnel()
    Dim i As Integer, j As Integer
    'Définit le tableau à 2 dimensions ainsi que leur
taille.
    Dim VarTab(1 To 3, 1 To 6) As String


    For i = 1 To UBound(VarTab, 1) 'boucle sur la 1ere
dimension
        For j = 1 To UBound(VarTab, 2) 'boucle sur la
2eme dimension
            'Alimente les éléments du tableaux
            VarTab(i, j) = i & j
            'Lit les éléments du tableau
            Debug.Print VarTab(i, j)
        Next j
    Next i
End Sub
```

Le code suivant alimente chaque élément du tableau avec des lettres aléatoires, entre A et Z.
Ensuite la macro trie par ordre croissant une des colonnes (au choix de l'utilisateur), dans la 1ere dimension du tableau.

Vba
```
Option Explicit

Sub TriCroissantMulticolonnes()
'Déclare un tableau à 2 dimensions.
Dim Tableau(1 To 4, 1 To 50) As String
Dim i As Integer, j As Integer, y As Integer
Dim indexColTri As Byte
Dim t As Variant
Dim Resultat As String

'-----Chargement du tableau -----------------
'Remplit chaque élément avec des lettres aléatoires,
entre A et Z
For i = 1 To UBound(Tableau, 2)
    For j = 1 To UBound(Tableau, 1)
        Randomize
        Tableau(j, i) = Chr(Int((26 * Rnd) + 1) + 64)
```

```
    Next j
Next i
'-----------------------------------------

'---- Applique un tri sur une des colonnes du tableau -
'Choisissez la colonne à trier: 1= 1ere colonne , 2=
2eme colonne...etc ...
indexColTri = 1

'On sort si l'index de colonne indiqué (indexColTri)
est plus grand que la taille de la première
'dimension dans le tableau.
If indexColTri > UBound(Tableau, 1) Then Exit Sub

For i = 1 To UBound(Tableau, 2)
     For j = 1 To UBound(Tableau, 2) - 1

        '---------
        'syntaxe pour le tri de données type Date
        'If CDate(Tableau(indexColTri, j)) >
CDate(Tableau(indexColTri, j + 1)) Then
        'Pensez à adapter le type de variable: Dim
Tableau(1 To 4, 1 To 50) As Date

        'syntaxe pour le tri de données type numérique
        'If CDec(Tableau(indexColTri, j)) >
CDec(Tableau(indexColTri, j + 1)) Then
        'Pensez à adapter le type de variable: Dim
Tableau(1 To 4, 1 To 50) As Long ...

        'syntaxe pour le tri de données type Texte
        If Tableau(indexColTri, j) >
Tableau(indexColTri, j + 1) Then
        '---------

            For y = 1 To UBound(Tableau, 1)
                t = Tableau(y, j)
                Tableau(y, j) = Tableau(y, j + 1)
                Tableau(y, j + 1) = t
            Next y

        End If
    Next j
Next i
'-------------------------------------------------------

'---- Affiche le résultat dans la fenêtre d'exécution -

For i = 1 To UBound(Tableau, 2)
    Resultat = ""

    For j = 1 To UBound(Tableau, 1)
        Resultat = Resultat & Tableau(j, i) & vbTab
    Next j

    Debug.Print Resultat
Next i
'-------------------------------------------------------

End Sub
```

3. L'instruction ReDim

3.1. ReDim

L'instruction **ReDim** est utilisée pour définir (ou redéfinir), en cours de procédure, l'espace mémoire alloué à un tableau dynamique.

ReDim sert pour:

- Redéfinir le nombre d'éléments.
- Changer le nombre de dimensions.
- Etablir les limites supérieures et inférieures de chaque dimension.

Exemple:

Vba
```
Option Explicit
Option Base 0

Sub Test()
    'Déclare la variable tableau
    Dim NomTableau() As Single
    Dim i As Integer

    'Redéfinit la taille du tableau
    ReDim NomTableau(2)
    'Boucle sur les éléments du tableau pour le remplir
    For i = 0 To UBound(NomTableau)
        NomTableau(i) = (3 + i) / 2
    Next i

End Sub
```

Vous pouvez appliquer l'instruction **ReDim** plusieurs fois dans une même procédure.

A chaque fois que vous modifiez la taille d'un tableau, le contenu des anciens éléments est effacé.

Vba
```
Option Explicit
Option Base 0

Sub Test()
    'Déclare la variable
    Dim NomTableau() As String
    Dim i As Integer

    'Définit la taille du tableau
    ReDim NomTableau(5)
    'Boucle sur les éléments du tableau pour le remplir
    'avec les lettres A,B,C,D,E et F
    For i = 0 To UBound(NomTableau)
        NomTableau(i) = Chr(65 + i)
    Next i

    'Renvoie la lettre A
    MsgBox "Premier élément du tableau: " &
NomTableau(0)

    'Redéfinit et réduit la taille du tableau.
    ReDim NomTableau(3)

    'Renvoie une chaîne vide: les anciens éléments ont
été effacés
    'lorsque la taille du tableau a été redéfinie.
    MsgBox "Premier élément du tableau: " &
NomTableau(0)
End Sub
```

Attention à la saisie du nom de la variable tableau lorsque vous utilisez **ReDim**. L'instruction a une action déclarative si la variable spécifiée n'existe pas formellement. Même si **Option Explicit** est ajouté en tête de module, une erreur de saisie dans le nom ne renverra pas de message d'erreur et un nouveau tableau sera créé.

Vba

```
Option Explicit

Sub Test()
   Dim NomTableau() As Long

    ReDim NomTablau(5)
    'Le "e" a été oublié volontairement pour montrer que contrairement
    'aux variables classiques (scalaires), la procédure ne renvoie pas d'erreur, même
    'si option Explicit à été déclaré: Un deuxième tableau est créé.

End Sub
```

Lorsque vous appliquez **ReDim** sur des tableaux formalisés, vous pouvez modifier le nombre d'éléments mais pas directement le type de données. Si vous souhaitez aussi changer les types de données en cours de procédure, utilisez une variable Variant et la syntaxe suivante:

Vba
```
Sub Test()
   Dim NomVariable As Variant
    Dim i As Integer

    'Modifie la variable en tableau de type String.
    ReDim NomVariable(1 To 3) As String
    'Alimente les éléments en données String
    For i = 1 To UBound(NomVariable)
        NomVariable(i) = Chr(64 + 1)
    Next i

    'Modifie la variable en tableau de type Integer.
    ReDim NomVariable(1 To 2) As Integer
    'Alimente les éléments en données numériques
    For i = 1 To UBound(NomVariable)
        NomVariable(i) = i
    Next i

End Sub
```

3.2. Le mot clé Preserve

Nous avons vu que l'instruction **ReDim** modifie la taille des tableaux, mais efface les anciens éléments.
Ajoutez le mot clé **Preserve** pour agrandir un tableau dynamique tout en conservant les valeurs existantes. Vous pourrez ainsi modifier la taille de la dernière dimension d'un tableau sans perdre les données déjà stockées dans les éléments d'origine.

Cet exemple montre comment lister dans un tableau à deux dimensions, le nom des fichiers d'un répertoire et leur date de création ou de dernière modification. Comme le nombre de fichiers n'est pas connu à l'avance, la taille du tableau augmente d'une unité à chaque tour de boucle, sans effacer les enregistrements qu'il contient déjà.
Vous remarquerez que c'est la dernière dimension du tableau (variable x) qui est modifiée.

Vba
```
Option Explicit

Sub ListeFichiersRepertoire()
    Dim Repertoire As String, Fichier As String
    Dim Tableau() As Variant
    Dim x As Integer, i As Integer
    Dim VerifTab As Variant

    'Définit le répertoire pour la recherche
    Repertoire = "C:\Documents and Settings\dossier"
    'Recherche tous les types de fichiers
    Fichier = Dir(Repertoire & "\*.*")

    'Boucle sur les fichiers pour récupérer les infos
    Do While Fichier <> ""
        'Incrémente le compteur de fichiers
        x = x + 1

        '--- Redéfinit la taille de la dernière dimension du tableau
        ReDim Preserve Tableau(1 To 2, 1 To x)
        '-----------------------------------

        'Récupère le nom du fichier
        Tableau(1, x) = Fichier
        'Récupère  la date et l'heure de création ou de dernière modification.
        Tableau(2, x) = FileDateTime(Repertoire & "\" & Fichier)
        Fichier = Dir
    Loop

    '--- On vérifie si le tableau est vide
    On Error Resume Next
    'VerifTab va prendre la valeur Empty si le tableau est vide.
    VerifTab = UBound(Tableau)
    On Error GoTo 0

    If IsEmpty(VerifTab) Then Exit Sub
    '---

    'Boucle pour lire le contenu du tableau.
    'UBound(Tableau, 2) permet de récupérer la limite supérieure de la 2eme dimension
    For i = 1 To UBound(Tableau, 2)
        'Inscrit le résultat dans la fenêtre d'exécution (Ctrl+G)
        Debug.Print Tableau(1, i) & " --> " & Tableau(2, i)
    Next i
End Sub
```

Le mot clé Preserve:

- Permet uniquement de modifier la limite supérieure de la dernière dimension du tableau.
- Ne permet pas de modifier le nombre de dimensions.

4. Option Base

La limite inférieure des tableaux est définie par l'instruction **Option Base**. La valeur peut être 0 ou 1.

La base par défaut est 0 si l'instruction n'est pas spécifiée dans le module.

Option Base doit être placée tout en haut du module, avant toute procédure ou déclaration.
Celle ci est valable uniquement pour le module où elle est située.

Vous pouvez vérifier l'action d'**Option Base** en testant les deux codes suivants.
Le premier (base 0) renvoie des limites 0 et 5. Le tableau peut donc contenir 6 éléments.

Vba
```
Option Explicit
Option Base 0
```

```
Sub Test_Base()
    Dim NomTableau(5) As String

    MsgBox "Index inférieur: " & LBound(NomTableau) & vbCrLf & _
        "Index supérieur: " & UBound(NomTableau)
End Sub
```

Le deuxième (base 1) renvoie des limites 1 et 5. Le tableau peut donc contenir 5 éléments.

Vba
```
Option Explicit
Option Base 1

Sub Test_Base()
    Dim NomTableau(5) As String

    MsgBox "Index inférieur: " & LBound(NomTableau) & vbCrLf & _
        "Index supérieur: " & UBound(NomTableau)
End Sub
```

Pour vous affranchir des particularités d'**Option Base**, vous pouvez aussi utiliser la clause To, afin de contrôler la plage des indices d'un tableau.

La syntaxe est: NomTableau(LimiteInférieure To LimiteSupérieure).
```
Dim NomTableau(1 To 5) As String
```

Pour définir un tableau multi dimensionnel:
```
Dim NomTableau(1 To 5, 1 To 20) As String
```

Cet exemple renvoie des limites 1 et 5:

Vba
```
Option Explicit

Sub Test_Base()
    Dim NomTableau(1 To 5) As String

    MsgBox "Index inférieur: " & LBound(NomTableau) & vbCrLf & _
        "Index supérieur: " & UBound(NomTableau)
End Sub
```

5. Les fonctions LBound et UBound

Les fonctions **LBound** et **UBound** permettent de déterminer la taille d'une dimension dans un tableau.
Leur principe d'utilisation est identique.

5.1. LBound

LBound Renvoie le plus petit indice disponible pour la dimension indiquée.

```
MsgBox LBound(NomTableau, 2)
```

Ici, la procédure affiche la limite inférieure de la 2eme dimension.
Utilisez LBound(NomTableau, 3) pour la 3eme dimension ...etc...

1 sera la valeur par défaut si l'argument dimension n'est pas spécifié. Pour tester la première dimension (ou un tableau à dimension unique) vous pouvez donc écrire:

Vba
```
Option Explicit
Option Base 0

Sub Test_LBound()
    Dim NomTableau() As Single

    ReDim NomTableau(8)

    'Affiche la taille inférieure d'un tableau à taille unique.
    'LBound renvoie 0 car "Option Base 0" est indiqué en tête de module.
    MsgBox LBound(NomTableau, 1)
    'ou plus simplement:
    MsgBox LBound(NomTableau)
End Sub
```

La limite inférieure d'une dimension peut être:

- 0 ou 1, en fonction de la valeur de l'instruction Option Base (Consultez le chapitre **Option Base** pour plus de détails).
- N'importe quelle valeur pour les dimensions définies à l'aide de la clause **To**.

LBound provoque une erreur si les dimensions des tableaux n'ont pas été initialisées.

Vba
```
Option Explicit
Option Base 0

Sub Test_LBound()
    Dim Tab_x() As Long
    Dim Tab_y(1 To 20, 5 To 30) As Integer
    Dim Tab_z(10) As String

    ReDim Tab_x(5)
    Debug.Print LBound(Tab_x) 'Renvoie 0

    Debug.Print LBound(Tab_y) 'Renvoie 1
    Debug.Print LBound(Tab_y, 2) 'Renvoie 5

    Debug.Print LBound(Tab_z) 'Renvoie 0

    ReDim Tab_x(4 To 8, 1 To 10, 1 To 20)
    Debug.Print LBound(Tab_x) 'Renvoie 4
    Debug.Print LBound(Tab_x, 3) 'Renvoie 1
End Sub
```

5.2. UBound

UBound Renvoie l'indice le plus élevé disponible pour la dimension indiquée.

```
MsgBox UBound(NomTableau, 2)
```

Ici, la procédure affiche la limite supérieure de la 2eme dimension.
Utilisez UBound(NomTableau, 3) pour la 3eme dimension ...etc...

1 sera la valeur par défaut si l'argument dimension n'est pas spécifié. Pour tester la première dimension (ou un tableau à dimension unique) vous pouvez donc écrire:

Vba
```
Option Explicit
Option Base 0
```

```
Sub Test_UBound()
    Dim NomTableau() As Single

    ReDim NomTableau(8)

    'Affiche la taille inférieure d'un tableau à
dimension unique.
    'UBound renvoie 8
    MsgBox UBound(NomTableau, 1)
    'ou plus simplement:
    MsgBox UBound(NomTableau)
End Sub
```

UBound provoque une erreur si les dimensions des tableaux n'ont pas été initialisées.

D'autres tests pour la fonction **UBound**:

Vba

```
Option Explicit

Option Base 0

Sub Test_UBound()
    Dim Tab_x() As Long
    Dim Tab_y(1 To 20, 5 To 30) As Integer
    Dim Tab_z(10) As String

    ReDim Tab_x(5)
    Debug.Print UBound(Tab_x) 'Renvoie 5

    Debug.Print UBound(Tab_y) 'Renvoie 20
    Debug.Print UBound(Tab_y, 2) 'Renvoie 30

    Debug.Print UBound(Tab_z) 'Renvoie 10

    ReDim Tab_x(4 To 8, 1 To 10, 1 To 20)
    Debug.Print UBound(Tab_x) 'Renvoie 8
    Debug.Print UBound(Tab_x, 3) 'Renvoie 20
End Sub
```

Retrouvez l'article complet de SilkyRoad en ligne : Lien61

Comment déployer vos applications professionnelles développées avec Microsoft Access 2007 en incluant le Runtime

Ce document a pour but de simplifier la compréhension du déploiement des applications développées avec Microsoft Access 2007. Il concerne Microsoft Office 2007 Prof+® associés à Microsoft Access Developer Extensions for Office 2007® en exploitant l'utilisation du Runtime.

1. Avant propos

Ce document a pour objectif de satisfaire aux demandes des développeurs ayant besoin de déployer des applications développées avec Microsoft Access 2007. Deux autres tutoriels pour les versions 2000 ou 2002 (XP) (Lien62) et 2003 (Lien63) sont disponibles sur le forum.

Vous allez le remarquer pour la plupart d'entre vous, l'approche et le mode de déploiement des applications issues d'Office 2007 est différente bien qu'une certaine similitude avec l'outil d'empaquetage de Microsoft Access Developper 2003.
Une évolution intéressante est en place : l'empaquetage se génère désormais depuis le Menu Microsoft Office

Retrouvez toutes les étapes du déploiement d'applications professionnelles Microsoft Access 2007 décrits par argyronet en ligne : Lien64

La FAQ Access

C'est quoi un twips ?

C'est une unité de mesure utilisée par le VBA (et Access) qui correspond à 1/1440 pouces soit 2,54cm
Ce qui fait qu'un cm contient à peu près 567 twips

Cette unité de mesure est complètement dissociée de la notion de pixel.
Son objectif est de permettre un positionnement impeccable d'une fenêtre (au sens global du terme... cela englobe donc les contrôles, etc.) sur un écran, et ceci, quel que soit la résolution de ce dernier.

Une application développée sous Access est-elle compatible avec toutes les versions d'Access ?

Lire une application avec une version d'Access supérieure ou égale à celle utilisée lors du développement ne devrait pas poser de problème.
Cependant, l'inverse n'est pas vrai. A savoir qu'une base développée sous Access 2003 ET enregistrée au format Access 2003 ne sera pas exploitable par les versions antérieures d'Access.

Qu'est-ce qu'un fichier mde ?

Un fichier Mde est une base de données Access stockée dans un état compilé. La modification des formulaires, états, macros et modules y est impossible. La création d'un fichier Mde améliore les performances de l'application, réduit sa taille et protège sa partie applicative des modifications. Attention, lorsque vous convertissez votre base en Mde, gardez précieusement une copie du fichier Mdb originel car la compilation est irréversible.

Que signifie VBA?

VBA pour Visual Basic édition (ou en Anglais : "for") Application (càd. orienté vers les applications bureautiques de Microsoft, Access, Excel, Word, ...).

Certains éléments qu'on peut être amené à gérer dans une application en Access ou en Excel, par exemple, sont identiques. D'autres sont spécifiques.

VBA peut être vu comme une restriction de VB. Mais l'interface de développement d'Access, par exemple, apporte pas mal de facilités.

Retrouvez ces Q/Rs sur la FAQ Access : Lien65

Les derniers tutoriels et articles

Présentation de la barre d'outils développeurs pour Internet Explorer

Tutoriel de présentation de la nouvelle barre d'outils pour développeurs Web pour Internet Explorer

1. Introduction

Le marché des navigateurs Web est actuellement dominé par Internet Explorer qui se fait peu à peu rattraper par Mozilla Firefox. Firefox a su utiliser les faiblesses d'Internet Explorer pour se faire adopter par une grande partie des utilisateurs. Ces faiblesses étaient(sont encore pour certaines) d'abord le mauvais respect des spécifications du W3C et la limitation quant aux fonctionnalités du navigateur.
Microsoft a su écouter ses utilisateurs en sortant une nouvelle version (Internet Explorer 7) respectant bien mieux les spécifications Web (XHTML, CSS, etc) et étant plus ouverte à l'ajout d'add-ins. Parmi les add-ins connus pour Internet Explorer 7, nous avons l'add-in Web Developper Toolbar (le sujet de cet article) mais également IE7 Pro dont Baptiste Wicht fait une présentation ici ainsi que IE Plus qui fera l'objet d'un prochin article.

Les développeurs Web d'aujourd'hui sont souvent soit pro-IE (pour sa forte part de marché) soit pro-Firefox (pour son respect des spécifications du W3C) (Lien66) et développer un site se révèle parfois compliqué car il est nécessaire de tester le site Web sous chaque navigateur.
Est alors arrivé firefox et son extension IETab (Lien67) qui permet de tester un site web sous Internet Explorer sans quitter Firefox. Après cela est apparu l'extension pour Firefox: Web developper (Lien68) qui propose un très grand nombre d'options, de validation de code, d'aide à la création de CSS, et bien d'autres. Il était alors plus que logique de délaisser Internet Explorer pour Firefox avec les deux extensions précitées, installées...

Jusqu'à la sortie de l'extension "IE developper toolbar". Celle-ci est la version IE de l'extension "Web developper". Nous allons donc au cours de cet article voir si elle peut remplacer sa "concurrente".

2. Présentation

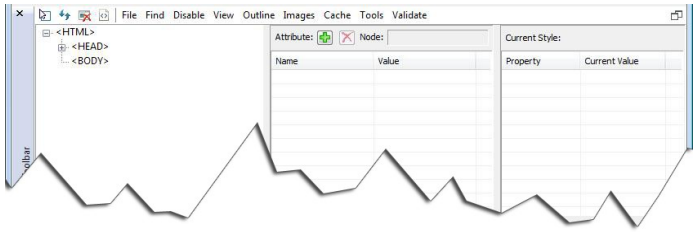
La première chose à faire consiste à télécharger l'extension pour Internet Explorer 7 (Mais qui devrait aussi fonctionner sur IE6) à l'adresse suivante : Téléchargement (Lien69).
Une fois l'extension installée, lancez Internet Explorer puis cliquez sur l'icône de la barre d'outils "IE developper toolbar". Cette icône est peut-être cachée; il vous faut dans ce cas cliquer sur la flèche à droite du menu Outils.

Cette extension propose de nombreuses fonctionnalités dont voici une liste (non exhaustive):

- Explorer et modifier la structure d'une page web en live - DOM (Document Object Model).
- Localiser et sélectionner des éléments spécifiques sur une page Web à travers une variété de techniques.
- Désactiver de façon sélective sélectives des paramètres d'Internet Explorer.
- Voir les noms des objets HTML comme les classes, les ID, et tout autre sorte d'attributs comme les touches d'accessibilité, etc
- Faire ressortir les tables, les cellules, les images ou les tags sélectionnés.
- Valider le code HTML, CSS, WAI (Accessibilité), et les liens de flux RSS.
- Afficher les dimensions des images, la taille des fichiers, chemin de l'image et le texte alternatif.
- Redimensionner la fenêtre du navigateur pour simuler différentes résolutions.
- Possibilité de vider le cache ou effacer les cookies (ou alors seulement ceux associés à un domaine spécifique).
- Afficher des règles pour aligner et/ou mesurer précisément les objets sur vos pages Web.
- Trouver les règles de style utilisées sur un élément spécifique de la page web.
- Voir le code source de façon formatée et colorée.

3. Fonctionnalités

Certains d'entre vous ne savent peut-être pas à quoi correspondent les fonctionnalités listées et je vais en donc en décrire certaines plus en profondeur afin qu'ils comprennent l'intérêt, **pour un développeur web**, de ne pas se passer de cette extension.
Tentons donc de vous faire apprécier ses fonctionnalités.
Voici l'aperçu général qui contient un menu et trois panneaux possédant chacun son intérêt.



3.1. L'utilisation de DOM

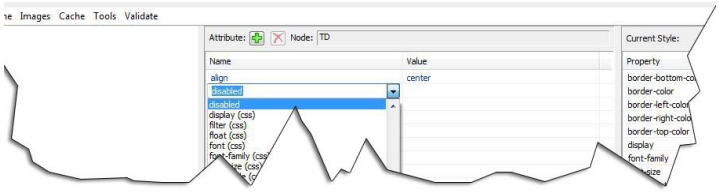
DOM (Document Object Model) est une technologie qui fait partie du Web depuis pratiquement ses tout débuts. DOM est une conception de la structure d'une page permettant de manipuler ses éléments, ou si vous préférez, un arbre hiérarchique des éléments du pages (images, paragraphes, tableaux, cellules, etc) et qui

permet de parcourir les éléments de cet arbre jusqu'à un élément donné et d'en modifier les valeurs si nécessaire. DOM est le partenaire d'AJAX dans un grand nombre de cas pratiques.
Je doute néanmoins que vous utilisiez cette barre d'outils afin de créer du code AJAX. Son utilisation est plutôt destinée au code (x)HTML notamment pour la création de feuilles de style, en sachant par exemple où se trouve tel ou tel élément et quels sont ces attributs comme son ID.

3.1.1. Edition de l'arbre

La première chose intéressante est la modification de l'arbre DOM ou si vous préférez, l'édition de la page en temps réel mais sans rien changer sur le serveur. Tout se passe en local et vous pouvez tester toutes les manipulations de votre choix sans le moindre risque. L'add-in est pour cela très facile d'utilisation et même un webmestre débutant y retrouvera ses marques.

Une fois sur votre page, dans la partie la plus à gauche, vous dépliez l'arbre jusqu'à trouver l'élément qui vous intéresse. Une fois cliqué dessus, s'affichent dans la partie centrale les attributs de cet élément et c'est ces derniers que vous pouvez modifier. Vous n'avez qu'à cliquer sur une case pour choisir parmi tous les attributs reconnus puis en lui indiquant une valeur. La modification se fera alors instantanément dans la page Web.



3.1.2. Information sur les éléments HTML de la page

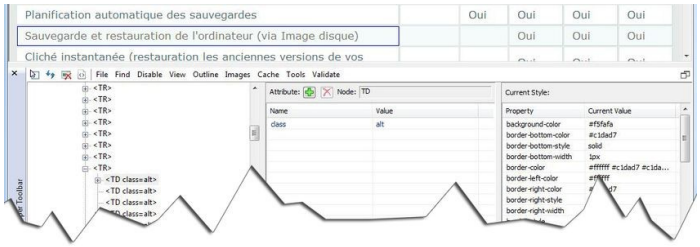
Cette fonctionnalité est peu souvent utilisée mais quand elle l'est, elle est d'une grande aide. Son principe: mettre en évidence certains éléments de la page. Simplement via les menu View et Outline vous pourrez faire ressortir des éléments comme certains attributs spécifiques ou alors les éléments DIV (c'est les paramètres que j'ai choisi sur la capture suivante:

3.2. Les feuilles de style CSS

Il est impossible aujourd'hui de créer un bon site web professionnel ou personnel sans l'utilisation des feuilles de style, qui permettent grâce à un seul fichier, de personnaliser l'affichage, tant les couleurs que la position de chaque élément dans la page.
L'extension n'a pas pour but de vous aider à écrire une feuille de style, mais plutôt à déboguer la feuille lorsqu'un problème d'affichage se présente ou alors, il vous sera possible sur un autre site web, de simplement pointer un élément avec la souris pour voir s'afficher tous les règles de style qui s'y appliquent. Pour cela, il vous faut savoir toutes les règles de style qui s'appliquent à cet élément et en faire la liste en analysant le code HTML et le fichier CSS peut se révéler très difficile, encore plus si la page utilise plusieurs fichiers CSS.

3.2.1. Création / Déboguage de CSS

Il peut arriver que vous vouliez copier le style d'une page web ne vous appartenant pas ou tout simplement que vous vouliez savoir pourquoi l'affichage d'un élément ne correspond pas à ce que vous souhaitiez. Pour cela, cliquez sur le menu **Find > Select Element by click** puis passez votre souris au dessus de la page Web. A chaque fois que votre curseur rencontrera un élément (une image, une cellule, un titre, etc.), ce dernier sera entouré par une bordure bleue afin de vous indiquer l'élément marqué, puis vous devez alors cliquer une fois avec le bouton gauche de la souris. La partie inférieure se met alors à jour. Vous avez dans la partie gauche le positionnement de votre élément dans l'arbre DOM, ce qui peut vous aider pour l'héritage ainsi que la visualisation des styles dits "inline". La partie centrale montre également les attributs de chaque élément mais permet avant tout d'en ajouter à la volée (pour les tests) ou d'en modifier une valeur.
Enfin, la partie de droite vous permet de voir toutes les règles qui s'appliquent à votre CSS. Cela vous permet entre autres de voir que parfois des règles s'appliquent alors que vous ne le voudriez pas ou que, au contraire, une règle que vous attendiez, ne s'applique pas. Dans ces deux cas, cela vient toujours d'une erreur humaine et c'est seulement grâce à ce genre de fonctionnalités, qu'il est possible de trouver rapidement le problème.



3.2.2. Nettoyage des feuilles de style

Une dernière fonctionnalité qui peut se révéler fort intéressante consiste à nettoyer votre feuille de style de toute règle d'affichage non utilisée, ce qui rend le fichier plus léger et plus facile à lire.
Pour faire cela, cliquez sur le menu **View > CSS Selector Matches**, ce qui aura pour conséquence de lister les règles CSS et d'indiquer le nombre d'éléments "touchés" par cette règle. Lorsqu'une règle n'est pas utilisée (0 match(es) for), vous pouvez la supprimer de votre feuille de style.

3.3. L'accessibilité

Lorsque vous créez un site web, vous devez penser à l'affichage qu'il aura sur différents navigateurs et donc penser également aux navigateurs moins connus, principalement dédiés aux personnes ayant un handicap. Vous aurez aussi les navigateurs "mobiles" comme sur le PDA ou téléphones mobiles (mais c'est une autre spécification) ainsi que les navigateurs en mode texte et/ou à lecture audio (Text-To-Speech). Pour tout cela, il existe différentes initiatives dont les plus connues reste le groupe de travail WAI (Lien70) et la "section 508" (Lien71), qui définissent les points à suivre pour développer un site web accessible à tous. Il est important de noter que développer un site qui ne serait pas pleinement accessible à des personnes ayant un handicap comme les aveugles, s'apparente à de la discrimination et est illégal dans

de nombreux pays. Il faut savoir qu'à peine 5% des 100 sites les plus visités au monde sont pleinement compatibles et respectent tous les conseils d'accessibilité.

En résumé cela consiste à avoir un document structuré (bonne utilisation des headers), des textes alternatifs pour les liens et les images, ainsi que des touches d'accès pour naviguer parmi le site web.

Nous allons maintenant voir que l'extension peut justement nous aider à respecter ces conseils afin de rendre votre site web pleinement accessible.

Cela peut se faire par deux principales fonctionnalités: la première, c'est la validation des règles d'accessibilité. Il vous suffit de cliquer sur le menu Tools > accessibilité et de choisir parmi les sites normes existantes (WAI et norme 508)

La seconde solution est de se mettre à la place d'un utilisateur avec un écran limité ou même un navigateur limité (voire en mode texte pour les aveugles. Texte uniquement car un logiciel tiers se charge alors de lire la page Web à haute voix).

Ainsi vous pouvez grâce à cette extension, désactiver les images ou le style CSS, changer la taille de l'écran pour simuler une résolution particulière ou encore afficher les accessKeys pour voir ce que pourra faire l'utilisateur avec un handicap.

4. Conclusion

On ne peut qu'apprécier l'effort utile réalisé par les développeurs de Microsoft en développant une extension qui permet aux développeurs Web de développer plus rapidement son site Web tout en restant sous Internet Explorer. Dans le pire des cas, si celui-ci aime à développer sous Firefox, avoir cette extension lui permettra de tester déboguer sous IE et de trouver plus rapidement les soucis sous le célèbre navigateur.

De mon avis personnel, Firefox est et restera mon navigateur par défaut mais force est de constater que Internet Explorer s'améliore de jour en jour et devient alors un vrai outil plutôt qu'un simple navigateur. Vivement la version 8 avec un éventuel moteur d'extensions.

Retrouvez l'article de Louis-Guillaume Morand en ligne : Lien72

L'actualité Windows

Microsoft Windows XP durera plus longtemps que prévu

Suite aux problèmes que rencontre Windows Vista pour s'imposer sur le marché et remplacer Windows XP. Microsoft a décidé de repousser la date de fin de Windows XP pour laisser le temps aux gens de migrer sous Vista. Par la date de fin, j'entends la date à laquelle Windows XP ne sera plus vendu.

La date de fin pour XP est donc repoussée de 6 mois jusqu'en Juin 2008. Cela fait maintenant déja 7 ans que XP a vu le jour et il montre encore une fois sa force.

Espérons que Microsoft utilisera ce temps supplémentaire pour améliorer son nouvel OS et surtout sa compatibilité avec les logiciels actuels sur le marché.

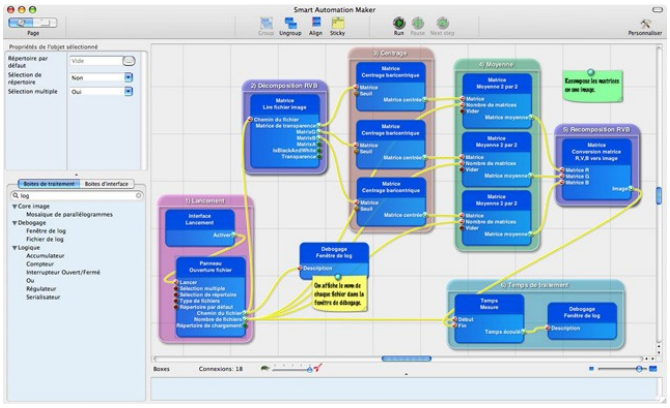
Retrouvez ce billet sur le blog de Baptiste Wicht : Lien73

Les derniers tutoriels et articles

Sébastien Marchand: Tour d'horizon d'Objective-C/Cocoa avec SAM

Développeur Cocoa depuis quelques temps déjà, j'ai eu l'occasion de constater que le langage Objective-C est assez peu connu et encore moins le framework Cocoa utilisé sous OSX. J'espère que ce petit tour d'horizon avec mon projet SAM (Smart Automation Maker) vous mettra l'eau à la bouche et donnera envie à certains de tenter la découverte de Cocoa.

1. Qu'est-ce que SAM ?

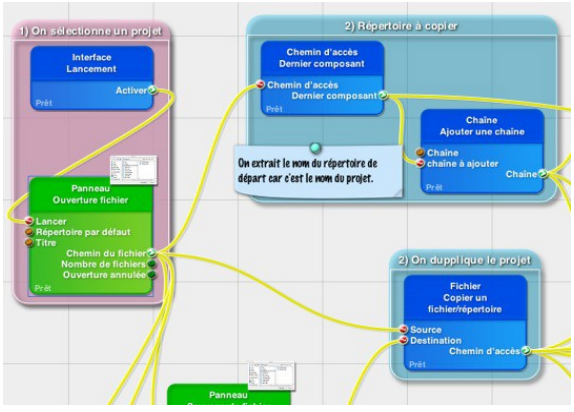


Smart Automation Maker est un logiciel hybride à mi-chemin entre Quartz Composer (QC) (Lien74), Interface Builder (IB) (Lien75) et Automator (Lien76). Il permet de développer des applications sous OS X de manière graphique en interconnectant des boîtes entre elles à la manière de QC (bien que la gestion des flux soit un peu différente). Il est possible de développer ses propres boîtes via un assistant. Ce projet est un produit commercial qui sera disponible prochainement.

2. Présentation des technologies Objective-C/Cocoa employées pour ce projet

2.1. Dessins Vectoriels et bitmaps

Cocoa permet une grande liberté en matière de dessin vectoriel. Il est ainsi possible de dessiner des objets vectoriels complexes (NSBezierPath), de dessiner des splines, de gérer des effets d'ombrage (NSShadow) ainsi que la transparence ou bien encore d'y mixer du texte (NSString) et des images bitmap (NSImage). OS X gérant d'entrée de jeux l'antialiasing le rendu est pour le moins convaincant

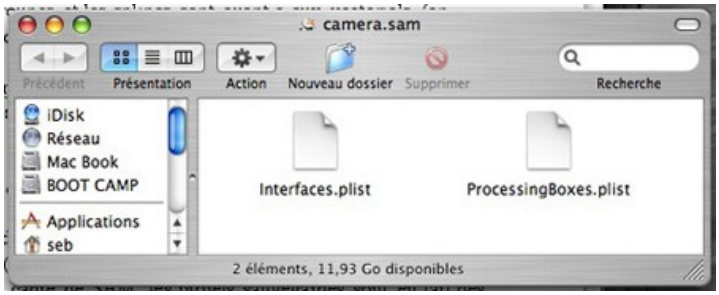


Dans l'exemple ci-dessus, les boîtes vertes et bleues sont dessinées sous forme de bitmap antialiasé en fonction du zoom. Les groupes et les splines sont quant à eux vectoriels (on notera au passage l'effet de transparence et d'ombrage sur les groupes) La classe NSImage utilisée ici pour les images bitmap est capable de lire une multitude de formats et même de nombreux types de fichiers RAW (format brut des appareils photos numériques). On saisit donc tout l'impact d'iPhoto sur la richesse de cette classe.

2.2. Gestion des fichiers de ressource

L'une des particularités d'OS X vient du fait que les applications Cocoa sont en fait des répertoires (des Bundles) à part entière et non des binaires bruts. On peut ainsi y stocker toutes formes de données allant d'un simple fichier de configuration texte à des images ou du son. Cela s'applique aussi pour les frameworks (les « DLL » de Cocoa) ou bien tout fichier construit à partir de données hétéroclites. On peut ouvrir un paquet en faisant un clic droit sur le fichier -> Afficher le contenu du paquet...

Ci-dessus, dans le cadre de SAM, les projets sauvegardés contiennent en fait deux fichiers XML (l'un pour l'interface graphique et l'autre pour la partie traitement).
Ce principe est aussi utilisé pour la génération des auto-exécutables de SAM. L'application est alors constituée d'un bundle d'application "type" codé pour lire les fichiers XML dans ses ressources et les plugins des boîtes utilisées par le projet y sont aussi copiés pour un fonctionnement autonome.

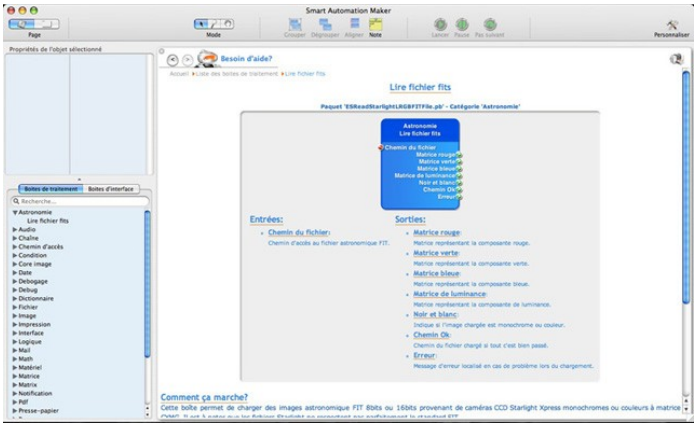


2.3. Internationalisation

OS X contrairement à Windows est 100% multilingue quelle que soit sa version. Cela se retrouve aussi au niveau des applications Cocoa. Les différentes traductions (fichier InfoPlist.strings pour le texte brut et fichier nib pour les interfaces graphiques générées par Interface Builder) sont alors stockées dans le répertoire de l'application comme des ressources à part entière. Dès lors, on comprend qu'il est très facile de traduire une application sans même disposer du projet d'origine. Ce fonctionnement est aussi exploité par SAM pour une traduction Français/Anglais (voir démo de création d'une boîte donnée en lien plus bas

2.4. Gestion des plugins

L'ensemble des boîtes de traitement ou d'interface de SAM sont autant de plugins (NSBundle) qui viennent enrichir les fonctionnalités du logiciel. Cocoa fournit toutes les interfaces nécessaires à leur chargement dynamique. A cela, s'ajoute toutes les possibilités de gestion des ressources et d'internationalisation déjà évoquées. Il en découle que chaque plugin intègre sa propre documentation multilingue. L'accès à la documentation se fait par une interface web (WebView) intégrée à l'application. Ici ce sont les technologies Web de Safari qui sont mises à contribution.



Besoin d'imprimer ? Pas de problème, l'accès à l'impression du système est disponible avec toutes les vues (terme employé pour définir une zone d'affichage dans l'interface). On peut même sauvegarder la page web en PDF (les API d'Aperçu sont passées par là...).

2.5. Mutli-threading

Particularité de SAM, chaque boîte est indépendante afin d'exploiter au mieux les machines multi-coeurs et/ou multi-cpu. Il est possible d'exécuter toute méthode d'une classe (on conserve ainsi l'accès aux variables de la classe) en détachant un thread (NSThread). Les données échangées peuvent être « facilement » synchronisées (NSLock) pour sécuriser les accès concurrents.

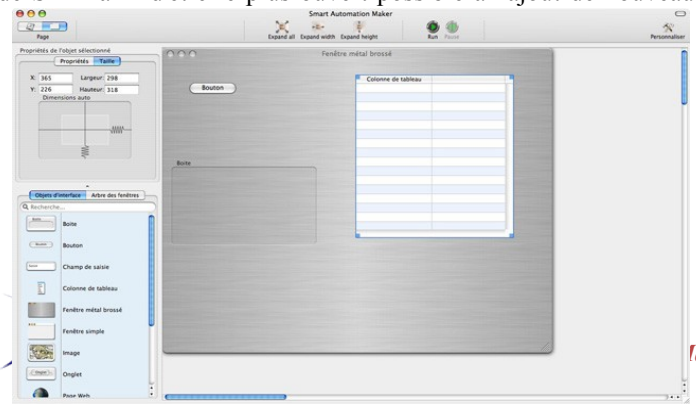
2.6. Sérialisations/Copies

L'ensemble des classes usuelles se conforme aux protocoles de copie (NSCopying/NSMutableCopying) et de sérialisation (NSCoding). L'écriture et la duplication de données s'en trouvent simplifiées. SAM se base par exemple sur la sérialisation pour l'écriture des projets sur disque sous forme de fichiers XML plist. On retrouve ici le fonctionnement d'iTunes et d'iPhoto pour sauvegarder la structure de leur librairie.

2.7. Capacités d'introspection et de mutation

Objective-C est un langage dynamique par définition. Il fournit ainsi de puissants mécanismes d'introspection mais aussi la possibilité d'enrichir des classes existantes (usage des Catégories) lors de l'exécution.

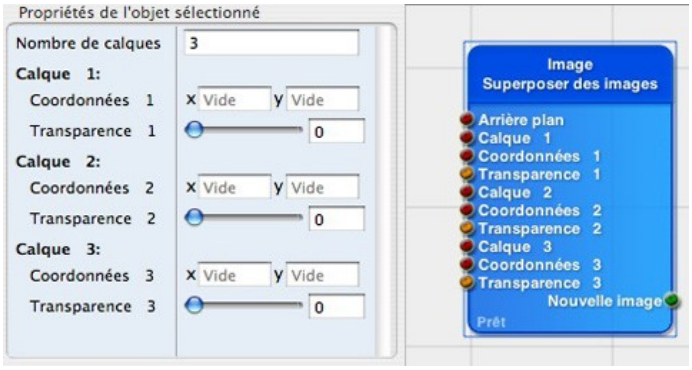
L'introspection est massivement utilisée dans l'éditeur d'interface de SAM afin d'être le plus ouvert possible à l'ajout de nouveaux composants graphiques inconnus.



Les catégories sont quant à elles utiles pour enrichir des classes comme des NSMutableArray ou bien des NSMutableDictionary afin d'y intégrer à la volée des sécurités (NSLock) pour l'accès concurrentiel des threads.

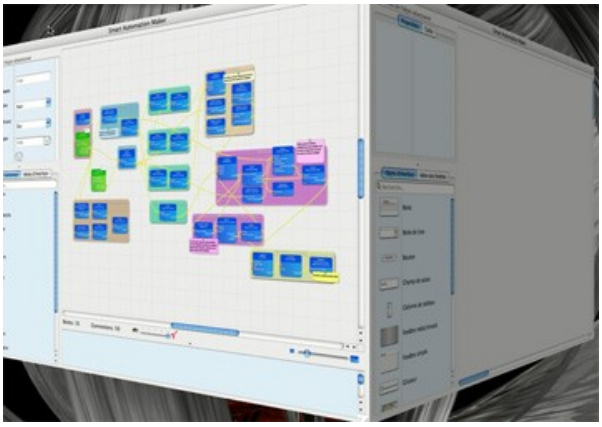
2.8. Notifications

Les notifications permettent la communication inter-classes (NSNotificationCenter) mais aussi inter-thread et inter-process (NSDistributedNotificationCenter). Une boîte dont le nombre d'entrées est variable (l'utilisateur veut par exemple une boîte « superposer des images» avec 3 images au lieu de 2) peut ainsi se mettre à jour et demander à l'éditeur de propriétés de se mettre à jour en conséquence.

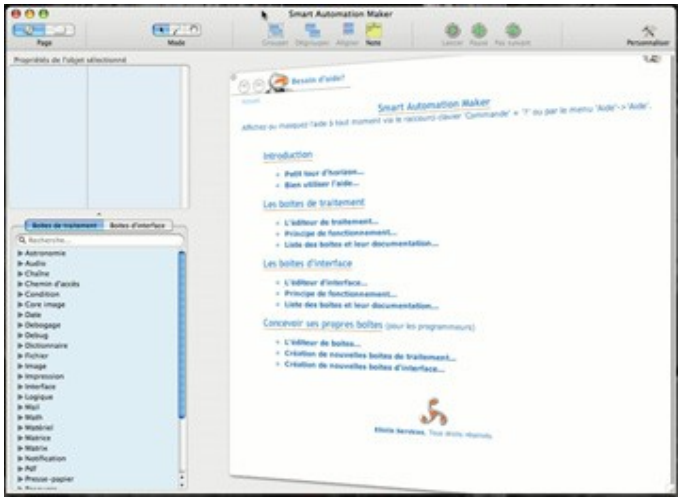


2.9. Core Image / Core Vidéo

L'arrivée d'OS X Tiger en Avril 2005 a été l'occasion pour Apple de mettre à disposition des programmeurs de nouvelles fonctionnalités telles que Core Image et Core Vidéo. SAM intègre ces technologies aussi bien au niveau des boîtes de traitements que des effets visuels 3D de l'interface. Le passage de l'éditeur de traitement à l'éditeur d'interface se fait ainsi avec un effet de cube 3D aussi utilisé dans Tiger lors d'un changement rapide de compte utilisateur.



La documentation est quant à elle située au verso de chaque plan de travail qui se retourne par pivotement (on retrouve d'ailleurs cet effet de transition dans le logiciel Keynote d'Apple).



3. Conclusion

Voilà pour une introduction non exhaustive au couple Objective-C/Cocoa sur un cas concret. J'espère avoir donné envie à ceux qui ne connaissent pas encore Objective-C/Cocoa de sauter le pas.

Pour plus d'informations sur SAM et le voir à l'oeuvre (vidéos de démo QuickTime), faire un tour sur : www.sam.eliotis.com (Lien77)
Les plus intéressés (personne ne dort?) pourront découvrir la conception d'une boîte de traitement avec l'assistant (Lien78). On y voit notamment l'intégration avec Xcode ainsi que la gestion multilingue de Cocoa.

4. Liens

- Objective-C : Lien79
- Cocoa : Lien80

Pour plus d'informations sur les classes présentées au long de cet article, je vous invite à consulter la documentation officielle : Lien81

Retrouvez l'article de Sébastien Marchand en ligne : Lien82

La FAQ MAC - RealBasic

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 ?") ;
- la classe AutoDiscovery pour créer des applications qui s'auto-détectent sur le réseau.
- et bien évidemment les prix changent aussi.

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.

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.

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
```

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.

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.

Retrouvez ces Q/Rs sur la FAQ MAC : Lien83

Hardware

Les derniers tutoriels et articles

Installation d'outils de gestion de sources (SCM) sur un NAS

Comment installer Trac et Subversion sur un NAS DS-101 ? Telle est la question que je me suis posée récemment.

1. Introduction

Quand je me suis acheté un NAS il y a quelques années déjà, j'ai choisi un Synology DS101 (maintenant remplacé par les 107, 207 ou 407 selon le nombre de disques durs que l'on veut) car le code source était sous GPL. Naturellement, le fabriquant a mis des protections, mais au fur et à mesure, des outils sont apparus pour débrider la bête (et maintenant, le fabricant autorise officieusement ce genre de pratiques). Un autre avantage du DS101 par rapport à d'autres NAS est qu'il n'a pas de ventilateur et est donc silencieux (à l'époque, le disque était tout de même dans ma chambre), malheureusement, le disque interne est en IDE. Maintenant, les DSx0x possèdent des ventilateurs et supportent uniquement les disques SATA.

Récemment, j'ai découvert Trac (Lien84) et son intégration avec Subversion (Lien85), idéal pour développer, mais il faut un serveur dédié, et ne voulant pas de nouvelle machine consommatrice d'électricité, je me suis retourné vers mon NAS. C'est ce que je vais décrire dans cet article, du débridage à la configuration des outils.

2. Accès au système du NAS

Par défaut, il n'y a aucun accès au système d'exploitation, pas de SSH, pas de telnet. Mais dès le départ, cette protection a été contournée et maintenant il est très facile d'activer telnet sur le NAS (il vaut mieux avoir la dernière version du firmware pour être sûr que Synology a bien supprimé le mot de passe Synology qui varie selon les jours). A l'aide des fichiers disponibles ici (Lien86), il est possible d'activer ou de désactiver telnet à l'aide de l'interface de mise à jour du NAS sur le serveur Web.

Une fois l'opération effectuée, il est possible de se connecter en root, le mot de passe étant le mot de passe administrateur de la machine.

3. Installation des outils de gestion des paquets

Les Synology tournent sur deux plateformes, ARM et PPC, selon les versions. Il faut alors utiliser un boostrap pour installer un gestionnaire de paquets. Les fichiers sont ici pour les ARMs (Lien87) et ici pour les PPC (Lien88). Ce fichier doit être copié sur le Synology puis exécuté grâce à telnet.

Pour les derniers NAS de Synology, la série DS-x07, il n'y a pas encore de bootstrap car les processeurs ARM ne fonctionnent plus dans le même mode que pour les précédents. Il faut donc encore un peu patienter.

Une fois installé et redémarré, il est possible que les services ne redémarrent pas (Apache et Samba). Le disque n'est plus accessible, la configuration Web déclenche une erreur. Cela est dû à une ligne manquante dans /etc/ld.so.conf : il faut y ajouter /lib. Une fois ajouté, il faut lancer la commande ldconfig et redémarrer le serveur.

4. Configuration du serveur Subversion

Apache 2 vient directement avec un module DAV SVN. L'avantage en outre est que le serveur est directement lancé au démarrage sur le port 8000 (le port 80 est réservé au serveur Apache natif du système).

Il faut tout d'abord installer les paquets adéquats :

```
ipkg svn
ipkg apache2
```

Une fois installé, on peut créer un repository n'importe où (par exemple sur le disque accessible de l'extérieur, donc sur /volume1/svn entre autres) et configurer comme on le désire ce repository.

Après, il faut démarrer le serveur et ajouter une ligne dans le fichier /opt/etc/apache2/httpd.conf afin de lui ajouter la configuration du module DAV :

```
Include /opt/etc/apache2/conf.d/mod_dav_svn.conf
```

Pour plus de renseignements sur la configuration du serveur, je vous renvoie au tutoriel dédié à Subversion (Lien89) pour adapter le fichier mod_dav.svn.conf.

L'intérêt de mettre le repository sur un dossier accessible de l'extérieur est qu'il sera facile de modifier ainsi les fichiers de configuration.

Une fois le repository paramétré, il faut redémarrer le serveur pour l'activer.

Dans le fichier de configuration d'Apache httpd.conf, une grande liste de modules est chargée au démarrage du service. Ils ne sont pas nécessaires et sont désactivables (il suffit de commenter les lignes LoadModule).

Il faut changer les propriétés du repository pour que l'utilisateur nobody puisse y accéder en écriture. Le plus simple est donc de lui donner la propriété des dossiers ainsi qu'au groupe users. Ainsi, en permettant un accès en lecture/écriture à l'utilisateur et au groupe, le serveur Apache pourra modifier le repository ainsi que vous depuis l'explorateur de fichiers.

5. Installation et utilisation de TRAC

5.1. Installation des modules nécessaires

TRAC est un module Python qui peut être exécuté par plusieurs modules, comme mod_fastcgi qui s'interface avec plusieurs serveurs, ou mod_python ou mod_wsgi. Il faut tout de même commencer par récupérer les paquets TRAC :

```
ipkg install py25-trac
ipkg install svn-py
```

5.2. Initialisation de TRAC

Une fois installé, un environnement TRAC peut être créé :

```
trac-admin /volume1/trac/ initenv
```

Une série de questions sont posées, la seule importante est le chemin vers le repository Subversion qui n'est pas le bon par défaut (si vous désirez une autre base de données, pas la peine d'installer py-sqlite).

Il est possible d'utiliser TRAC directement à l'aide de tracd :

```
tracd --port 800 /volume1/trac/
```

5.3. Chargement automatique de TRAC au démarrage

Trac ne peut malheureusement pas encore être chargé à partir d'Apache. La raison est que Python est buggé sur les DS (il n'est pas capable de gérer les flottants, ce qui est utilisé d'une manière ou d'une autre dans les mods Apache). Il faut donc charger tracd au démarrage en mode daemon.

A nouveau, l'utilisateur nobody devra avoir accès au dossier trac en lecture, en écriture et exécution.

Pour lancer tracd automatiquement au démarrage, le plus simple est d'utiliser un script de démarrage (Lien90) placé dans /opt/etc/init.d. Normalement, il y a un script S80Apache dans ce même dossier. Le but est de créer un fichier presque identique, mais qui lancera le daemon tracd.

L'option -d permet de lancer tracd en mode daemon

6. Communication entre TRAC et Subversion

Le principal avantage de Subversion est la possibilité d'avoir des post-commit hooks. Il s'agit d'un script exécuté après le commit sur le repository (il en existe d'autres qui peuvent se déclencher avant le commit pour une vérification par exemple des fichiers).

Dans le dossier hooks se trouvent des patrons pour ces scripts. Par exemple le script suivant appellera trac-post-commit-hook.py (Lien91) placé dans /volume1/trac :

post-commit

```
#!/bin/sh

 REPOS="$1"
 REV="$2"
 LOG=`/opt/bin/svnlook log -r $REV $REPOS`
 AUTHOR=`/opt/bin/svnlook author -r $REV $REPOS`
 TRAC_ENV='/volume1/trac'
 TRAC_URL='http://trac.mysite.com/project/'

 /opt/bin/python2.5 /volume1/trac/trac-post-commit-hook
\
   -p "$TRAC_ENV"  \
   -r "$REV"       \
   -u "$AUTHOR"    \
   -m "$LOG"       \
   -s "$TRAC_URL"
```

7. Conclusion

Même si les NAS de Synology ne permettent pas un fonctionnement correct de Python, il est possible de les utiliser pour la gestion de projet. Même s'il y a deux serveurs en réalité, Apache et mod_proxy peuvent rediriger une nouvelle page vers le serveur tracd.

Retrouvez l'article de Miles en ligne : Lien92

Quelques conseils pour bien choisir votre écran

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

1. Introduction

L'écran est un périphérique clé pour votre ordinateur. En effet, sans lui, vous ne pourriez rien afficher et il serait tout simplement impossible de communiquer avec votre ordinateur, ce qui le rendrait inutile.

On trouve plusieurs sortes d'écran et dans plusieurs tailles. Les prix varient fortement, c'est donc surtout le budget qui va vous limiter dans ce choix-là. Il faudra bien entendu choisir en fonction de vos besoins, mais le choix reste tout de même assez simple pour la majorité des cas. Il suffira souvent de définir la taille en fonction de votre budget. Néanmoins, pour les besoins spécifiques comme le jeu ou l'imagerie, le choix peut se révéler plus compliqué.

2. Type

Il existe plusieurs types d'écran :

- Ecran **cathodique (CRT)** : Ces écrans très encombrants fonctionnent avec des rayons d'électrons qui sont projetés depuis le tube vers des points multicolores qui forment ensemble une image sur l'écran.
- Ecran **LCD** : Plus souvent appelé tout simplement écran plat. Ce sont des écrans fins (voire très fins) qui fonctionnent avec un système à cristaux liquides.

Aujourd'hui, les écrans à tube sont remplacés par les écrans LCD, mais il reste encore un grand nombre d'écrans cathodiques. Néanmoins, il en reste très peu sur le marché. Et bien qu'au départ les taux de réponse étaient beaucoup plus élevés sur les écrans LCD, aujourd'hui, on peut trouver de bon temps de réponse ce qui est indispensable pour les joueurs. Mais les CRT supplante toujours les LCD en terme de performances dans les jeux. Je ne peux que vous conseiller d'opter pour un écran LCD qui prendra moins de place chez vous.

Néanmoins, les écrans LCD ont quelques désavantages par rapport aux CRT. Premièrement, il faut utiliser la résolution native de la dalle pour utiliser les performances optimales de l'écran. Et ensuite, les écrans LCD ont en général un moins bon rendu des couleurs et de l'image qu'un écran CRT.

3. Taille

La première chose que l'on regarde en achetant un écran, c'est bien évidemment sa taille, exprimée en pouces. La taille d'un écran peut faire de 15" à 26" en général, mais on trouve quelques écrans qui dépassent les 30 pouces. Ces derniers étant terriblement chers et réservés soit au visionnement de vidéos soit à l'affichage de cartes ou de graphiques. Plus l'écran est grand plus la résolution qu'il permet d'afficher est élevée. Avec un 22" par exemple, vous pouvez visionner 2 pages A4 Word En même temps à l'écran.

La norme actuelle est de 17 pouces et vous trouverez des écrans de cette taille à moins de 150 euros. On commence à trouver aussi de plus en plus d'écran 19" qui sont déjà très confortables et leur prix reste assez bas (entre 140 et 350 euros). Si vous avez le budget pour, je vous conseille de prendre directement un écran 19 pouces. Si vous regardez souvent des films sur votre ordinateur ou si vous jouez beaucoup, vous pouvez aller encore plus haut avec par exemple un 22" ou 24", mais la gamme de prix n'est plus la même (500 à 800 euros pour un 24").

Si vous comptez acheter un écran de très grande taille (30" par exemple), faites attention à ce que votre carte graphique puisse le supporter. En effet, pour afficher de telles tailles d'écran, il faut utiliser la technologie DVI Dual Link. Cette technologie permet d'afficher des résolutions très élevées en utilisant deux émetteurs TMDS sur un seul câble.

Il faut également faire très attention à la résolution que l'écran peut afficher. En effet, certains écrans de 19" n'affichent que 1024x768 alors que les écrans de cette taille affichent normalement 1280x1024, ce qui est de mon avis assez inadmissible. Voilà les résolutions conseillées en fonction de la taille de l'écran :

- 15" : 1024*768 en 4:3
- 17" : 1280*1024 en 4:3 et 1440*900 en 16:10
- 19" : 1280*1024 en 4:3 et 1440*900 en 16:10
- 20" : 1600*1200 en 4:3 et 1680*1050 en 16:10
- 22" : 1680*1050 en 16:10
- 24" : 1920*1200 en 16:10
- 30" : 2560*1600 en 16:10

En plus de la taille et de la résolution, on peut aussi éventuellement s'intéresser au format de l'écran. Il existe en effet différents formats :

- **4/3** : C'est le format "normal" pour un écran. Par exemple 1280*1024
- **16/9** : C'est un format large, comme au cinéma, beaucoup de contenus vidéo sont dans ce format. La résolution serait par exemple de 1280*720 ou 1366*768. Mais on trouve peu de vrai 16/9, comme vous le verrez ensuite.
- **16/10** : C'est le format qui vient souvent remplacé le 16/9. Pourquoi ? Tout simplement, parce qu'en informatique, les nombres comme 720 et 1366 ne sont pas vraiment aimés. C'est pourquoi les fabricants, utilisent par exemple 1280*768, ce qui équivaut à un format 16/10. La conséquence sera que vous aurez quand même des bandes noires quand vous regarderez du contenu 16/9. A noter que la plupart des jeux fonctionnent en 16/10, donc vous n'aurez pas toute la place occuppée par le jeu et le GPU devra travailler plus qu'avec une configuration normale.

Si vous comptez regarder beaucoup de films, je vous conseile d'acheter un écran 16/9 (ou 16/10), mais dans le cas contraire, un écran 4/3 sera très satisfaisant.

Faites attention au fait que plus la résolution est elevée, plus il y aura de pixels à dessinner et plus la carte graphique devra travailler. Si vous comptez opter pour un grand écran (+24"), je vous conseille d'avoir une carte graphique qui tient la route.
IV. Temps de réponse
Un paramètre important est le temps de réponse, surtout pour les joueurs. C'est le temps en millisecondes que met l'écran pour faire passer un pixel du blanc au noir. Si vous faites essentiellement de la bureautique, ce temps de réponse n'est pas très important, mais faites quand même attention à ce qu'il ne dépasse pas les 20 ms. Par contre, pour les joueurs, c'est un autre problème. En effet, avec un temps de réponse élevé, vous risquez d'avoir un effet de flou dû aux changements trop rapides pour l'écran. Dans ces conditions, jouer à certains jeux 3D peut vite se révéler impossible. C'est pourquoi, je vous conseille de prendre des écrans avec des taux de réponse de 5ms ou moins.

5. Dalle LCD

Si vous optez pour un écran LCD, vous serez confrontés à différents types de dalles :

- **TN** : Ce sont les dalles les plus rapides et les moins chères. Néanmoins, elles ont un angle de vision très faible et sont assez mauvaises pour la lecture de vidéos.

De plus, elles ne sont pas 8 bits, c'est à dire qu'elles n'arrivent pas au même nombre de couleurs que Windows (16.7 millions). Elles sont obligées d'avoir recours au dithering pour toutes les afficher, mais cela provoque un fourmillement désagréable à l'écran .Elles ne sont donc pas "True colors". Elles sont à réserver pour les joueurs ou les petits budgets.

- **VA (PVA et MVA)** : Ce sont les dalles les plus polyvalentes. La vitesse est moins rapide qu'avec une dalle TN, mais on arrive à des taux tout à fait raisonnables. Elles permettent de faire du jeu et de lire des vidéos confortablement. Elles sont un peu plus chères que les dalles TN.
- **IPS** : Ce sont les dalles avec le plus grand angle de vision. Bien qu'en retard jusqu'il y a peu sur les autres dalles, on trouve maintenant des dalles IPS avec des taux de réponses très corrects. Ce sont les meilleures dalles pour tout ce qui touche à la retouche vidéo.

Pour résumer, si vous cherchez le plus petit prix, utilisez une dalle TN et pareil si vous cherchez le plus petit taux de réponse. Si vous voulez un écran passe-partout, je vous invite à vous diriger vers une dalle MVA. Si vous faites exclusivement de la retouche photo, prenez une dalle IPS.

Néanmoins, on voit apparaître quelques nouveautés dans les dalles IPS qui les rendent de plus en plus compétitives, donc n'hésitez à comparer plusieurs types de dalles et ne vous arrêtez pas sur un seul type de dalle.

6. Luminosité/Contraste

Voici 2 autres paramètres assez importants pour votre écran. Commençons par la luminosité. Elle définit la visibilité de l'écran dans un environnement très éclairé et est calculée en candela par mètre carré. Ce n'est pas le paramètre le plus important, mais je vous conseille tout de même une luminosité minimale de 250cd/m2.

Passons au contraste qui est plus important que la luminosité. C'est la différence entre le point le plus lumineux et le plus sombre du moniteur. Plus le taux de contraste est élevé, plus le rendu des couleurs est bon. Je vous conseille un taux de contraste minimum de 500:1.

7. Autres

Une question que l'on n'a pas encore envisagée est le **multi-écrans**. Vous pouvez en effet avoir plusieurs écrans sur votre ordinateur. 2 avec des cartes graphiques à deux sorties et plus avec des cartes graphiques spéciales qui permettent de brancher un dédoubleur derrière les sorties de la carte. Cela vous permet de visualiser plusieurs choses en même temps. Par exemple, vous pouvez développer un programme et avoir devant vos yeux votre EDI et le résultat actuel du travail ou de la documentation. Ou alors plus simplement avoir votre playlist Itunes sur un écran et un jeu sur l'autre ou un jeu et internet ... Avec plus de deux écrans, c'est vraiment si vous avez besoin de toujours connaître l'état de certaines applications, mais c'est surtout au niveau professionnel que l'on utilise ça. Pour une simple utilisation, je vous conseille donc de rester avec un simple écran. Par contre, si vous voulez vous faire plaisir, un deuxième écran peut être un bon investissement.

Au niveau de la **connectique**, les écrans peuvent être raccordés soit par un câble DVI soit par un câble VGA. Certains écrans ont aussi une entrée haute-définition HDMI, mais c'est encore assez rare et ce n'est utile que si vous destinez votre écran à la visualisation de contenu HD. Si vous le pouvez, choisissez un écran avec une entrée DVI plutôt que VGA, car la prise VGA transporte de l'analogique, ce qu'il veut dire que la carte graphique doit transformer le numérique en analogique et l'écran fait ensuite de même à la récéption du signal. Cela peut se traduire par une perte de qualité. Certains écrans ont deux entrées et permettent de choisir la source, cela peut se révéler pratique si vous avez 2 PCs. Sur certains écrans, vous trouverez également un hub USB qui vous permet d'augmenter ainsi le nombre de ports USB de votre PC, cela peut-être pratique, car les ports sont ainsi assez facilement disponibles, mais il ne faut pas que ce soit un argument de choix.

On peut aussi parler du **rétro-éclairage**. La plupart des écrans actuels utilisent un rétro-éclairage à tube, mais certains écrans utilisent maintenant un rétro-éclairge à LED, ce qui permet un meilleur rendu des couleurs et une meilleure luminosité. Cette nouvelle technologie augmente néanmoins le prix de l'écran et est encore peu disponible sur les écrans de grande taille. De plus, il est encore difficile d'obtenir un éclairage harmonieux avec cette technologie, on a donc souvent les côtés plus clairs que le centre.

Certains écrans peuvent basculer du mode **paysage** au mode **portrait**. Vous pouvez ainsi adapter le mode avec l'activité que vous êtes en train de faire sur votre ordinateur. L'écran bascule donc d'un mode à l'autre et le driver graphique vous permet de changer l'affichage de votre ordinateur pour l'adapter au nouveau mode.

Il est aussi utile de regarder la **garantie** qui est proposée par le constructeur et/ou le fournisseur. Vous trouverez également parfois des garanties 0 pixel mort pendant x mois qui peuvent se révéler très intéressantes. De plus, faites bien attention à certaines clauses de la garantie, certaines garanties n'agissent que dès qu'il y a x pixels morts et pas avant.

Un argument que peu de personnes regardent mais qui peut intéresser certaines personnes, est "Est-ce que l'écran est prêt pour la **HD** ?". Pour ça, il faut voir les labels accordés à l'écran et faire attention à ces différents labels qui ne veulent pas dire la même chose :

- **Compatible HD** : Ce label ne permet pas d'afficher des vidéos en haute définition au contraire de ce que son nom indique. Il indique simplement que l'on peut lire une vidéo HD mais elle sera convertie dans un format standard. Vous pouvez donc la lire, mais comme une vidéo normale.
- **HD Ready** : Ce label indique une résolution d'au moins 720 lignes et un format 16/9. Vous pourrez donc voir vos vidéos en haute définition, par contre ce label ne garantit pas que l'image ne soit pas traitée, vous risquez donc une légère altération de la qualité d'images.
- **Full HD** : Ce label indique une résolution d'au moins 1080 lignes. C'est le nec plus ultra de la haute définition. Néanmoins, il faut faire attention au fait qu'il n'est pas certifié, la vigilance est donc de mise. Par contre les écrans avec ce label sont encore extrêmement chers.

Si vous voulez des vidéos en haute définition, je vous invite à éviter le label Compatible HD qui ne vaut pas grand chose et de vous orienter vers les formats HD Ready et Full HD.

Une dernière chose importante qu'il ne faut pas négliger est l'angle d'inclinaison possible soit vers le haut soit vers le bas. Plus ces angles sont élevées plus vous pourrez adapter l'écran à vos besoins et cela vous évitera des problèmes de cou. Il est également intéressant de considérer l'ajustement vertical possible.

C'est-à-dire la distance sur laquelle on peut monter ou descendre l'écran.

8. Conclusion

Pour résumer, la première chose à laquelle il faut penser est la taille d'écran que l'on veut et ensuite ce que l'on va en faire (jeux, 3D, HD ou encore bureautique), ce qui va définir les caractéristiques principales de la bête. Ensuite, on peut peaufiner les détails, toujours en fonction de nos besoins.

J'espère que mes conseils vous auront été utiles pour vous trouver un écran.

Deux petits liens qui pourraient en intéresser plus d'un :
- Les écrans Mac sont-ils True Colors ou pas ? : Lien93
- Un excellent topic sur les écrans LCD : Lien94

Retrouvez l'article de Baptiste Wicht en ligne : Lien95

Ruby/RoR

Les derniers tutoriels et articles

Tutoriel HAML

HAML est un langage de template pour Ruby on Rails qui se substitue au RHTML et apporte à Rails un vrai langage de template : clair, concis et puissant.

1. Introduction

HAML est un langage de template pour Ruby on Rails conçu pour créer des fichier HTML de manière rapide et élégante. Il se substitue au RHTML et apporte à Rails un vrai langage de template : clair, concis et puissant. De plus, HAML est très facile à prendre en main car il se base sur des langages déjà connus : HTML, CSS et Ruby. Comme dit son créateur, "HAML paraît bizarre les 20 premières minutes, mais après ça, vous irez plus vite"

Bien que ce tutoriel soit orienté sur Ruby on Rails, il est intéressant de noter que la portée de HAML dépasse le monde de Ruby et Rails. En effet, un moteur de template utilisant la syntaxe d'HAML existe depuis peu pour PHP, il s'agit de phpHaml (Lien96).

2. Installation

2.1. Installation du plugin pour Ruby on Rails

Le plugin HAML pour Ruby on Rails s'installe comme tout autre plugin. Il est téléchargeable à l'adresse http://haml.hamptoncatlin.com/download/ (Lien97).

Si vous débutez avec Ruby on Rails, voici la procédure à suivre pour installer ce plugin :
Placez-vous à la racine de votre application Rails, et entrez la commande suivante :

```
ruby script/plugin install
http://svn.hamptoncatlin.com/haml/tags/stable
```

Le script se charge alors de télécharger les fichiers nécessaires et les place dans le dossier vendor/plugins/.

Une fois le plugin pour Rails installé, vous pouvez commencer à écrire vos vues avec HAML. Pour l'utiliser à la place de ERB, il vous suffit de nommer votre fichier avec l'extension haml. Par exemple, le template de l'action inscrire du controlleur membres sera app/views/membres/inscrire.haml au lieu de app/views/membres/inscrire.rhtml. De plus, vous pouvez parfaitement écrire certaines vues en HAML et d'autres en RHTML, voire faire un partial en HAML et l'inclure dans une vue en RHTML et vice-versa. La transition entre RHTML et HAML est donc très facile, car elle peut se faire petit à petit. Sachez seulement que si les deux fichiers existent, c'est le fichier HAML qui sera utilisé.

2.2. Configuration de l'éditeur

HAML utilise une indentation de deux espaces pour sa mise en forme, il est donc important d'utiliser un éditeur bien configuré pour nous faciliter la tâche. Il existe actuellement des plugins pour les éditeurs suivants :
- TextMate
- Aptana/RadRails (Eclipse)
- JEdit
- (G)Vim
- Komodo
- Emacs

Ces plugins ont l'avantage d'ajouter également la coloration syntaxique, et quelques raccourcis qui vous faciliterons la tâche lorsque vous maîtriserez ce langage.

Vous pourrez retrouver les liens vers ces plugins sur http://groups.google.com/group/haml/web/syntax-highlighting (Lien98). Je me suis intéressé en particulier à TextMate et Aptana/RadRails pour vous proposer une méthode d'installation détaillée. N'hésitez pas à consulter la page des plugins pour installer le support de HAML dans votre éditeur favori.

2.2.1. Installation du plugin pour TextMate

Le bundle pour TextMate est disponible sur le repository SVN de l'éditeur. Pour le télécharger, lancez un Terminal, puis entrez les commandes :

```
mkdir -p /Library/Application Support/TextMate/Bundles
cd /Library/Application Support/TextMate/Bundles
svn co
http://macromates.com/svn/Bundles/trunk/Bundles/Haml.tmbundle
```

Si TextMate est lancé, vous pouvez utiliser la commande **Bundle > Bundle Editor > Reload Bundles** pour charger le plugin sans avoir besoin de relancer l'éditeur.

2.2.2. Installation du plugin pour Aptana/RadRails

Il existe un plugin pour Aptana/RadRails, l'éditeur gratuit basé sur Eclipse. Pour installer ce plugin :

- Allez dans le menu "Aide > Mises à jour de logiciels > Rechercher et installer" ...
- Sélectionnez "Rechercher les nouveaux dispositifs à installer" puis cliquez sur "Suivant"
- Cliquez sur "Nouveau site distant"
- Saisissez les informations suivantes :
- Nom : Plugin pour HAML
- Adresse URL : http://haml.lucky-dip.net/
- Cochez alors la case correspondant à la ligne créée, et cliquez sur Terminer

Aptana devrait trouver la dernière version du plugin et vous proposer de l'installer.

3. Un premier aperçu de HAML

Si vous connaissez déjà relativement bien le HTML, et a fortiori si vous avez déjà travaillé sur des templates RHTML, vous verez rapidement que HAML est un langage très simple et naturel. C'est d'ailleurs ce qui en fait son succès : son apprentissage peut prendre moins de 20 minutes pour les développeurs web confirmés.

Dans la tradition de Rails "Show don't tell", nous allons commencer par quelques exemples de conversion d'un template RHTML en HAML, tirés d'applications réelles. Ces exemples devraient simplement vous permettre de voir à quoi ressemble un template HAML. Ne vous inquiétez pas de ce que vous ne comprenez pas, nous verrons tous les détails de la syntaxe par la suite.

Pour avoir tout de même une idée de ce qui se passe, dites-vous simplement que :
- HAML reflète la structure du document par son identation (comme YAML)
- il mélange Ruby, HTML et CSS dans un tout cohérent

Le premier exemple est une page affichant une News sur un site sur lequel j'ai eu l'occasion de travailler. C'est l'exemple typique d'une page simple qui affiche un modèle provenant d'une base de données.

RHTML:

```
<% if @article.allow_comments? or @article.published_comments.size > 0 -%>
  <a name="comments"></a><h4 class="blueblk">Comments</h4>
  <% unless @article.comments_closed? -%>
    <p class="postmetadata alt">
      <small><a href="#respond"><%= _("Leave a response")%></a></small>
    </p>
  <% end -%>
  <ol class="comment-list" id="commentList">
  <% if @article.published_comments.blank? %>
    <li id="dummy_comment" style="display: none"></li>
  <% else %>
    <%= render(:partial => "comment", :collection => @article.published_comments) %>
  <% end %>
  </ol>
<% end -%>
```

HAML:

```
- if @article.allow_comments? or @article.published_comments.size > 0
  %a{:name => "comments"}
  %h4.blueblk Comments
  - unless @article.comments_closed?
    %p.postmetadata.alt
      %small= link_to _("Leave a response"), '#respond'

  %ol.comment-list#commentList
  - if @article.published_comments.blank?
    %li#dummy_comment{:style => "display: none"}
  - else
    = render(:partial => "comment", :collection => @article.published_comments)
```

Ce deuxième exemple est un extrait de Typo, le célèbre moteur de blog. C'est un morceau de code qui affiche les commentaires publiés pour un billet.

Le dernier exemple est un layout typique d'une application Rails, conçu directement en HAML :

```
!!!
%html( :xmlns => "http://www.w3.org/1999/xhtml" )
  %head
    %meta( 'http-equiv' => "Content-type",  'content' => "text/html; charset=utf-8" )
    %title= "Admin : #{controller.controller_name}"
    = stylesheet_link_tag 'admin'
    = javascript_include_tag :default

  %body
    #header
      %h1= link_to "Admin", admin_home_path
    #sidebar
      = render :partial => 'shared/sidebar'
    #contenu
      #flash= display_flashes
      #main= yield
```

Maintenant que vous avez eu un bref aperçu de ce à quoi peut ressembler un template HAML, voyons ensemble plus en détail la syntaxe du langage.

4. La Syntaxe

4.1. Les éléments principaux : faire du HTML

4.1.1. Les balises

Le caractère %, placé en début de ligne, désigne une balise :

```
%p
%li
%br
```

Les attributs sont notés sous la forme d'un Hash ruby :

```
%meta{'content-type' => 'text/html', 'charset' =>
'utf8'}
%label{:for => 'user_login'}

%ul{:id => 'menu'}
%p{:class => 'details'}
```

On retrouve les attributs 'id' et 'class' très souvent, on a donc introduit une syntaxe particulière, qui vient tout simplement de CSS :

4.1.2. Un peu de CSS

Placé après une balise, le # indique la valeur de l'attribut id :

```
%ul#menu     =>     <ul id="menu">
```

Placé après une balise, le . indique la valeur de l'attribut class :

```
%p.details     =>     <p class="details">
```

Avec la syntaxe # ou . il n'est pas obligatoire de donner le nom de la balise. Si on l'omet, HAML utilisera un 'div' :

```
#sidebar     =>     <div id="sidebar">

.footer      =>     <div class="footer">
```

Comme en CSS, on peut mélanger . et # ; On peut également définir plusieurs classes en les séparant par des . :

```
#colonne.gauche              =>     <div id="colonne"
class="gauche">

#messages.colonne.gauche     =>     <div id="messages"
class="colonne gauche">
```

On obtient alors une syntaxe très compacte et proche de celle utilisée en CSS

4.2. Interpréter du code Ruby

4.2.1. Afficher du contenu dynamique

Le caractère = placé soit en début de ligne, soit après une balise, désigne du code Ruby qui sera évalué et affiché (comme <%= ... %> avec RHTML) :

```
= @article.titre       =>     Introduction au moteur de
template HAML

%p= @article.titre     =>     <p>Introduction au moteur
de template HAML</p>
```

Bien évidement, on n'est pas limité aux simples variables. On peut utiliser n'importe quel code Ruby imaginable, en particulier les Helpers existant dans Rails.

```
%p= link_to "Accueil", accueil_path     =>     <p><a
href="/">Accueil</a></p>
```

4.2.2. Tests et boucles

Le signe - indique du code Ruby qui sera évalué mais ne sera pas affiché (comme <% ... %> avec RHTML). N'importe quel code Ruby peut être utilisé dans un template, mais on utilise cette syntaxe la plupart du temps pour des boucles ou des tests :
- if logged_in?
  Bienvenue dans votre espace personnel
- else
  Accès refusé

```
%ul
  - @articles.each do |article|
    %li= article.title
```

Il n'y a jamais de 'end', contrairement à ERb

4.3. Remarques

L'imbrication des balises et des blocs se fait simplement en fonction de l'indentation. Il n'y a pas de 'end' ou de balises fermantes avec HAML. L'indentation en HAML est aussi importante que le code, car elle définit la portée des élements. Ainsi vous pouvez changer facilement et rapidement l'imbrication des balises, car tous les éditeurs textes ont un moyen d'indenter ou désindenter les lignes selectionnées. En revanche, cela peut vous jouer des tours si vous n'y faites pas attention.

L'indentation est toujours de deux espaces, il ne faut jamais utiliser de tabulation. Faites bien attention sur ce point, car des erreurs d'indentation peuvent vous donner des messages d'erreur difficiles à interpréter.

C'est pour cette raison que bien configurer votre éditeur est primordial !

```
<div id="content">
  <div class="left column">
    <h2>Bienvenue dans notre site !</h2>
    <p>
      <%= print_information %>
    </p>
  </div>
  <div class="right column">
    <%= render :partial => 'sidebar' %>
  </div>
</div>
```

L'équivalent de ce code HTML sera :

```
#content
  .left.column
    %h2 Bienvenue dans notre site !
    %p= print_information
  .right.column= render :partial => "sidebar"
```

5. Aller plus loin

5.1. Exercez-vous !

Avant d'aller plus loin, prenez quelques minutes pour convertir un template d'une de vos applications. N'ayez pas d'inquiétude, si vous faites une erreur, il vous suffira de supprimer le fichier .haml que vous avez créé. Convertir quelques-uns de vos templates vous permettra de bien vous familiariser avec le langage. Dans le reste de cet article, vous trouverez quelques conseils et astuces pour vous aider à améliorer encore la lisibilité de vos vues.

5.2. Les balises inline

Un des premiers problèmes rencontrés lorsque l'on débute avec HAML est la gestion des balises inline, c'est-à-dire :
"Comment faire si je veux mettre un mot en gras ou en italique au milieu d'une phrase ?"
En HAML, la représentation suivant la structure du document serait :

```
%p
  Ceci est un paragraphe avec un mot en
  %strong gras
  et en
  %em italique
  \.
```

Ce n'est pas très lisible... Cependant dans ce cas, on peut écrire simplement :

```
%p
  Ceci est un paragraphe avec un mot en
<strong>gras</strong> et en <em>italique</em>.
```

Qui sera compilé en :

```
<p>
  Ceci est un paragraphe avec un mot en
<strong>gras</strong> et en <em>italique</em>.
</p>
```

De la même manière, pour interpréter des valeurs, plutôt que d'écrire :

```
%p
  Prix :
  = #{produit.prix}
  €
```

Il suffit de faire :

```
%p= "Prix : #{produit.prix}€"
```

On peut également utiliser un double égal qui évite l'utilisation de guillemets. Ce code ci-dessous est strictement équivalent au code précédent :

```
%p== Prix : #{produit.prix}€
```

On peut aussi utiliser les helpers, par exemple link_to pour faire un lien :

```
%p== Un lien vers la page #{link_to 'contact',
contact_url}.
```

5.3. Ecrire une expression sur plusieurs lignes

A l'inverse, vous aurez parfois besoin d'insérer du code sur plusieurs lignes. Pour cela, on utilise le caractère | à la fin de chaque ligne. Par exemple pour un helper particulièrement long :

```
%p= link_to_remote "Supprimer", |
  :url => {:action => 'delete_profile_from_duel', :id
=> 2}, |
  :before => "Element.show('wait_icon')", |
  :complete => "Element.hide('wait_icon')" |
```

Notez que cette syntaxe est un peu lourde, mais peut être très souvent évitée en utilisant les mécanismes de Rails. La vocation d'HAML est également de vous inciter à utiliser au mieux les Helpers et les routes nommées : si votre code n'est pas assez clair, refactorisez !

5.4. Attributs 'class' et 'id' dynamiques

Il n'est pas possible d'utiliser les raccourcis # et . pour spécifier dynamiquement un id ou une classe, il faut passer par un Hash dans lequel n'importe quel code Ruby est accepté :

```
- @items.each do |item|
  %tr{ :class => cycle('even', 'odd') }
    %td= item.titre
    %td= item.description
```

Si vous êtes adepte du plugin simply_helpful, vous serez heureux d'apprendre qu'il existe une syntaxe spéciale avec HAML pour ce plugin, il s'agit de [] qui remplace div_for :

Contrôleur
```
@article = Article.find(3)
```

Vue
```
%p[@article]     =>     <p id="article_3"
class="article">
```

5.5. Autres éléments

5.5.1. Doctype

Un triple point d'exclamation (!!!) insère un Doctype XHTML. Par défaut, le doctype "XHTML 1.0 Transitional" est utilisé, mais d'autres versions sont également supportées, ainsi que le prologue XML.

```
!!!
%html
  %head
    %title Utilisation de !!!
  %body
    %h1 Exemple de document HAML
```

Génère le document suivant :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html>
  <head>
    <title>Utilisation de !!!</title>
  </head>
  <body>
    <h1>Exemple de document HAML</h1>
  </body>
</html>
```

Voici la liste des doctypes disponibles :

```
!!!
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">

!!! strict
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">

!!! frameset
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
frameset.dtd">

!!! 1.1
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

!!! xml iso-8859-1
<?xml version='1.0' encoding='iso-8859-1' ?>
```

5.5.2. Commentaires

Le / crée un commentaire html : tout le contenu qui suit est englobé dans un commentaire :

```
/
  %p Ceci ne sera pas affiché ...
```

Sera compilé en :

```
<!--
```

```
  <p>Ceci ne sera pas affiché</p>
-->
```

Les commentaires conditionnels d'Internet Explorer peuvent être utilisés en les rajoutant juste derrière le /. Par exemple, pour inclure une feuille de style spécifique à Internet Explorer :

```
/[if IE]
  = stylesheet_link_tag 'ie'
```

Sera compilé en :

```
<!--[if IE]>
  <link href="/stylesheets/ie.css" media="screen"
rel="Stylesheet" type="text/css" />
<![endif]-->
```

5.5.3. Echappement

Le \ sert de caractère d'échappement : utile pour afficher une ligne qui commence par -, =, % …

```
%p
  \- Ceci ne sera pas interprété comme du code ...
```

Sera compilé en :

```
<p>
  - Ceci ne sera pas interprété comme du code ...
</p>
```

6. Conclusion

Vous devriez maintenant en savoir suffisament pour créer vos vues avec HAML. Si vous rencontrez des problèmes, n'hésitez pas à poser vos questions sur le forum Ruby on Rails (Lien99).

Pour conclure cet article, voici quelques liens intéressants :

- Le site officiel d'HAML : Lien100
- Le groupe de discussion sur Google Groups (en anglais uniquement) : Lien101

Retrouvez la suite de l'article de Thomas Brian en ligne : Lien102

Les livres Ruby/Ruby On Rails

Ruby et Rails - Le guide Ruby des développeurs Rails

Le guide Ruby des développeurs Rails.

Cet ouvrage est conçu pour aider les développeurs Rails à maîtriser Ruby. Chaque chapitre vous entraîne plus loin dans la compréhension de Ruby et met en évidence ses liens avec Rails. Vous apprendrez à travailler avec les objets et les classes ainsi qu'à exploiter la syntaxe élégante et expressive de Ruby pour développer des applications Rails performantes. Vous améliorerez vos qualités de développeur Rails grâce à une connaissance approfondie de Rails et apprendrez à mieux en tirer profit.

Les développeurs Ruby débutants trouveront dans ce livre une introduction simple au langage Ruby orienté Rails contenant des techniques de programmation dynamiques, une présentation des objets, des classes et des structures de données Ruby, ainsi que de nombreux exemples clairs de code Ruby et Rails à l'œuvre.

Critique du livre par Cécile Muno

Orienté entièrement développement, cet ouvrage est un plaisir pour le programmeur. Il évolue dans l'étude du langage Ruby puis de son Framework d'une manière intuitive. De la console à l'environnement de développement, on a le temps de "sentir" ce à quoi les concepteurs ont voulu aboutir en développant ce langage orienté web.
Un programmeur débutant dans ce langage ne doit pas avoir de crainte pour appréhender ce livre. Malgré tout, je ne conseille pas ce livre pour débuter l'apprentissage de la programmation, il y a des langages plus simples à maitriser auparavant.
En abordant le framework Rails, on entre vraiment au coeur du sujet. Il suffit dès lors de se laisser porter d'exemples en exemples, de manipulations en manipulations pour progresser à son rythme. Ce n'est pas un ouvrage qu'on lit d'un bout à l'autre d'une seule traite. Il faut appliquer progressivement les acquis à des situations concrètes ou à des idées à développer et toujours garder le livre sous la main comme tout bon dictionnaire.
Un seul petit reproche : les explications trop lapidaires pour l'installation de Ruby puis de Rails notamment dans les différentes manières de procéder et dans les "mauvaises" interactions existant avec d'autres environnements déjà installés sur votre ordinateur.

Conclusion : Un livre qui vous enseignera les bons réflexes de la programmation RoR tout en vous guidant vers les aspects les plus ardus du développement Web. Pour vous guider et vous aider dans votre apprentissage, surtout n'oubliez pas le forum, la FAQ et les tutoriels Ruby et Rails.

Retrouvez ce livre sur la rubrique Ruby/RoR : Lien103

Ruby on Rails

Le Framework qui révolutionne l'univers du développement Web

En l'espace de 18 mois, Rails a créé un véritable bouleversement dans le monde du développement Web, ralliant de nombreux développeurs PHP, Java ou. NET séduits par la cohérence et les gains de productivité offerts par ce Framework : plus de laborieux mapping objet-relationnel ni de fichiers de configurations multiples à gérer, un code concis et facile à faire évoluer, une intégration native d'Ajax qui permet de migrer sans douleur vers des interfaces Web 2.0 et, surtout, des projets qui se terminent dans les temps et vont au-delà des attentes des clients.
Une deuxième édition entièrement refondue du livre fondateur

Coécrit par David Heinemeier Hansson, le créateur de Rails, et Dave Thomas, le plus influent des experts de la communauté Ruby, cet ouvrage adopte une approche mêlant pratique et approfondissement technique, aussi efficace pour les néophytes que pour les développeurs ayant déjà une expérience de Rails.

La première partie propose un apprentissage par l'exemple, à travers la réalisation d'une application de commerce électronique complète. La deuxième partie de l'ouvrage décrit en profondeur toutes les composantes du Framework, la dernière partie présentant les meilleures pratiques en matière de sécurité et de déploiement.

Cette nouvelle édition s'est enrichie d'un an de retours d'expérience sur des projets de grande ampleur et détaille les

nombreuses nouveautés apportées par les versions 1.1 et 1.2 de Rails : les migrations, les templates RJS, l'intégration de Prototype, la gestion innovante des applications REST, les nouveaux outils de déploiement (Capistrano), sans parler des multiples améliorations apportées aux modules ActiveRecord, ActiveSupport et ActionPack.

Critique du livre par NoisetteProd

Curieux de découvrir le p'tit dernier dont on entendait tant parler, je me suis rendu dans ma librairie préférée pour chercher un livre qui me plairait sur Ruby on Rails. Mon choix s'est porté sur la deuxième édition du livre de Dave Thomas et David Heinemeier Hansson parmi une petite dizaine.

Mon choix s'est avéré judicieux ! J'ai dévoré d'une traite cet ouvrage en me régalant !

Pourquoi avoir choisi ce livre ? Plutot autodidacte, j'aime bien apprendre par la pratique et une partie entière de ce livre était une étude de cas qui semblait détaillée et que l'auteur nous aidait à mener a bien. Et en effet cette partie est, à mon goût, parfaite ! Une découverte progressive des concepts et du langage tout en répondant à une commande qu'un client aurait pu nous faire. L'auteur se permet même de jouer, en complicité avec le client, avec le Framework pour en présenter la flexibilité. Après chaque étape de progression dans la construction de ce projet, les auteurs proposent quelques minis exercices pour aller plus loin de ses propres ailes avec cependant, au travers de leur site anglais, des discussions sur les solutions à mettre en place. Cette étude est vraiment complète, des scénarios d'utilisation aux tests unitaires, digne d'un projet professionel.

Avant l'étude de cas, les auteurs ont pris la précaution, dans une première partie, de nous présenter ce qu'est une application Rails, comment installer l'environnement rails et un équivalent du fameux "Hello Word".

Après l'étude de cas, et malgré une sensation de parfaite maitrise de Rails, nous découvrons de manière minutieuse et pédagogique les entrailles du Framework.

La dernière partie, preuve de maturité du Framework et des retours d'expérience des auteurs, est consacrée à des astuces et bonnes manière de sécurité à appliquer dans les applications développée ainsi qu'aux techniques de déploiement, et de supervision de votre site web.

A noter que le langage Ruby n'est présenté qu'au travers d'une annexe assez "légère". Si vous découvrez Ruby, un support autre sera à prévoir.

Le plus difficile après avoir dévoré ce livre est de ne pas mettre en pratique immédiatement toutes les astuces et facilités qu'offre le Framework Rails !

Je ne saurais que vous conseiller ce livre si vous désirez découvrir Ruby on Rails !

Retrouvez ce livre sur la rubrique Ruby/RoR : Lien104

Pratique de Ruby on Rails

Le succès foudroyant de Ruby on Rails rappelle que les développeurs Web exigent dorénavant de leurs outils qu'ils s'installent aisément sur toutes les plates-formes, qu'ils minimisent le nombre de lignes de code à écrire (et à maintenir), et qu'ils offrent un environnement de test et de mise au point irréprochable.
Pour tirer le meilleur parti de Rails, il est indispensable de se familiariser avec Ruby, de connaître les règles qui président à la conception d'une base de données relationnelle, et de faire sien le fameux modèle MVC (Modèles, Vues, Contrôleurs). Éric Sarrion nous explique tout cela avec sa minutie habituelle, dans un ouvrage très progressif.

Pratique de Ruby on Rails se compose de 5 parties :

- La partie Introduction montre en 15 pages comment installer Rails et créer une petite application. Elle plonge ensuite dans les bases de la programmation Ruby, un langage qui s'avère particulièrement agréable à apprendre.
- La partie Contrôleurs montre comment écrire les actions destinées à être appelées par le serveur http. C'est là que le développeur décide de la réaction de l'application en fonction de l'url demandée.
- Dans la partie Modèles, vous apprendrez non seulement à créer et modifier les objets qui représentent votre base de données, mais aussi à concevoir une base de données relationnelle dans les règles de l'art.
- La partie Vues se concentre sur le rendu des valeurs calculées par les contrôleurs et les objets du modèle. C'est là que vous apprendrez à paginer votre affichage HTML, intégrer CSS et JavaScript, et bien sûr utiliser Ajax.
- La dernière partie aborde des fonctions indispensables comme l'envoi de mail ou la publication de services Web, sans lesquels une application web moderne ne serait pas complète.

Critique du livre par Pierre Chauvin

Pour être franc, je n'avais pas encore eu le temps de tester le moindre "Hello World !" avec Ruby ou RoR. C'est avec une certaine méconnaissance du langage et du framework que j'ai abordé le livre. Cet ouvrage est résolument tourné vers les développeurs débutants, le développeur expérimenté aura synthétisé la documentation du site officiel en quelques clics.

Après un léger tutoriel dédié à l'installation de Ruby, de Rails, vous réaliserez une application de type CRUD (Create, Read, Update, Delete) générée par Rails. Puis, le chapitre 4 condense les spécifications du langage et vous arme pour vos premiers programmes (SciTE Editor, les commentaires, variables et constantes, tests conditionnels et boucles, fonctions, classes, modules, collections, exceptions, etc.). Les parties II, III, IV décrivent ensuite le modèle Rails MVC: les contrôleurs, les modèles, les vues, le tout basé sur des exemples concrets. L'ensemble est expliqué simplement et de manière pragmatique. AJAX est également abordé, mais aussi l'envoi de mail SMTP, ou la publication et consommation de services Web, le paramétrage du serveur HTTP WEBrick (changement de port, les exceptions en production, la journalisation).

J'ai apprécié la vision adaptée pour une bonne compréhension des débutants, même si certaines explications sont un peu approximatives à mon gout (en POO notamment). Si vous souhaitez commencer le développement MVC avec Ruby ET Rails, ce livre accompagnera vos premiers pas !

Retrouvez ce livre sur la rubrique Ruby/RoR : Lien105

Conception

Les derniers tutoriels et articles

Tests unitaires et doublures de tests : les simulacres ne sont pas des bouchons

Le terme "objet simulacre" est devenu populaire pour décrire des objets spéciaux qui imitent de vrais objets dans le but de les tester. La plupart des environnements de développement ont maintenant des outils qui permettent de créer facilement des objets simulacres. Cependant, souvent on ne réalise pas que les objets simulacres sont un cas particulier d'objets de tests, qui permettent un style de test différent. Dans cet article j'explique comment les objets simulacres fonctionnent, comment ils encouragent le test basé sur la vérification du comportement, et comment la communauté autour d'eux les utilise pour développer un style de test différent.

Notes du traducteur

Le papier original en anglais est disponible ici (Lien106).

Le vocabulaire anglais employé dans la littérature de test est assez délicat à traduire en français (et je suis d'ailleurs soulagé que Martin Fowler souligne que même en anglais ce vocabulaire est assez confus). J'ai suivi les traductions proposées ici. Donc la traduction utilise la table de correspondance ci-dessous :

mot anglais	mot français
mock	simulacre
fake	substitut
stub	bouchon
dummy	fantôme

Il y a plusieurs alternatives : objets fantaisie, mimes, etc.
Le mot stub est particulièrement vide de sens pour un francophone... Cette discussion (Lien107) au sujet de la traduction de to stub out est assez révélatrice.

En ce qui concerne la traduction de l'acronyme TDD (pour Test Driven Development), l'équivalent français DDT (Développement Dirigé par les Tests) n'est pas très parlant, ni très courant. Par conséquent TDD n'a pas été traduit. De même pour BDD (Behavior Driven Development) dont l'équivalent français serait DDC (Développement Dirigé par le Comportement). BDD n'a pas été traduit non plus.

1. Introduction

J'ai rencontré pour la première fois le terme "objet simulacre" dans la communauté XP il y a quelques années. Depuis je les ai rencontrés de plus en plus. D'une part parce ce que beaucoup des principaux développeurs de ces objets ont fait partie de mes collègues à ThoughtWorks à différents moments. D'autre part parce que je les vois de plus en plus dans la littérature sur les tests influencée par XP.

Mais trop souvent je vois que les objets simulacres sont mal décrits. En particulier je les vois souvent confondus avec les bouchons - des utilitaires souvent employés dans les environnements de tests. Je comprends cette confusion - je les ai considérés comme similaires pendant un moment, mais des conversations avec les développeurs de simulacres ont fait régulièrement pénétrer un peu de compréhension des simulacres dans mon crâne de tortue.

Cette différence est en fait composée de deux différences distinctes. D'une part il y a une différence dans la vérification des tests : il faut distinguer vérification d'état et vérification de comportement. D'autre part il y a une différence de philosophie dans la manière dont le test et la conception interagissent, ce que je nomme ici par style "classique" et "orienté-simulacre" du développement dirigé par les tests.

(Dans la version précédente de cet essai, j'avais réalisé qu'il y avait une différence, mais je combinais les deux ensemble. Depuis, ma compréhension s'est améliorée et il est temps de mettre à jour cet essai. Si vous n'avez pas lu le précédent essai vous pouvez ignorer mes douleurs croissantes, car j'ai écrit cet essai comme si la précédente version n'existait pas. Mais si vous êtes familier avec la précédente version, vous pourriez trouver utile de noter que j'ai cassé la vieille dichotomie des tests basés sur l'état / tests basés sur l'interaction en deux dichotomies : celle de la vérification état/comportement et celle du style classique/orienté-simulacre. J'ai également ajusté mon vocabulaire pour qu'il corresponde à celui du livre de Gerard Meszaros xUnit Test Patterns).

2. Les tests traditionnels

Je vais commencer par illustrer les deux styles avec un exemple simple (l'exemple est en Java, mais les principes sont valables pour n'importe quel langage orienté-objet). Nous voulons prendre un objet Commande (Order) et le remplir à partir d'un objet Entrepôt (Warehouse). La commande est très simple, avec un seul produit et une quantité. L'entrepôt contient les inventaires de différents produits. Quand nous demandons à une commande de se remplir elle-même à partir d'un entrepôt, il y a deux réponses possibles. S'il y a suffisamment de produit dans l'entrepôt pour satisfaire la commande, la commande est considérée remplie, et dans l'entrepôt la quantité de produit est réduite du montant approprié. S'il n'y a pas suffisamment de produit alors la commande n'est pas remplie, et rien ne se produit au niveau de l'entrepôt.

Ces deux comportements impliquent quelques tests, qui sont des tests JUnit assez conventionnels :

```
public class OrderStateTester extends TestCase {
  private static String TALISKER = "Talisker";
  private static String HIGHLAND_PARK = "Highland
Park";
  private Warehouse warehouse = new WarehouseImpl();

  protected void setUp() throws Exception {
    warehouse.add(TALISKER, 50);
    warehouse.add(HIGHLAND_PARK, 25);
  }
  public void testOrderIsFilledIfEnoughInWarehouse() {
    Order order = new Order(TALISKER, 50);
    order.fill(warehouse);
    assertTrue(order.isFilled());
    assertEquals(0, warehouse.getInventory(TALISKER));
  }
  public void testOrderDoesNotRemoveIfNotEnough() {
    Order order = new Order(TALISKER, 51);
    order.fill(warehouse);
    assertFalse(order.isFilled());
    assertEquals(50, warehouse.getInventory(TALISKER));
  }
```

Les tests xUnit suivent une séquence typique de quatre phases : initialisation, exécution, vérification, nettoyage. Dans ce cas la phase d'initialisation est faite partiellement dans la méthode setUp (initialiser l'entrepôt) et partiellement dans la méthode de test (initialisation de la commande). L'appel à order.fill est la phase d'exécution. C'est là que l'objet est incité à faire la chose que nous voulons tester. Les déclarations assert sont ensuite la phase de vérification, elles contrôlent si la méthode exécutée a fait correctement son travail. Dans ce cas il n'y a pas de phase de nettoyage explicite, car le ramasse-miettes le fait implicitement pour nous.

Durant l'initialisation il y a deux sortes d'objets que nous mettons ensemble. Order est la classe que nous testons, mais pour que order.fill fonctionne, nous avons également besoin d'une instance de Warehouse. Dans cette situation Order est l'objet sur lequel nous focalisons le test. Les gens orientés-tests aiment utiliser des termes comme l'"objet en cours de test" ou le "système en cours de test" pour nommer une telle chose. Chacun de ces termes est difficile à dire, mais comme ils sont largement acceptés je me force à les utiliser. Suivant Meszaros j'utilise Système en cours de test, ou encore l'abréviation SCT.

Donc pour ce test j'ai besoin du SCT (Order) et d'un collaborateur (Warehouse). J'ai besoin de Warehouse pour deux raisons : d'une part pour faire fonctionner le comportement testé (puisque order.fill appelle les méthodes de warehouse) et d'autre part pour la vérification (puisqu'un des résultats de order.fill est un changement potentiel dans l'état de Warehouse). Au fur et au mesure que nous allons explorer ce sujet, vous allez voir que nous insisterons beaucoup sur la distinction entre le SCT et les collaborateurs (dans la version plus ancienne de cet article je parlais du SCT comme l'"objet primaire" et des collaborateurs comme les "objets secondaires").

Ce type de test utilise la **vérification de l'état**, ce qui signifie que nous déterminons si la méthode exécutée a fonctionné correctement en examinant l'état du SCT et de ses collaborateurs après l'exécution de la méthode. Comme nous le verrons, les objets simulacres permettent une approche différente de la vérification.

3.Les tests avec des objets simulacres

Maintenant je prends exactement le même comportement mais j'utilise des objets simulacres. Pour ce code j'utilise la bibliothèque jMock pour définir les simulacres. jMock est une bibliothèque java pour les objets simulacres. Il y a d'autres bibliothèques pour objets simulacres, mais celle-ci est à jour et est développée par les créateurs de cette technique, donc c'est un bon point de départ.

```
public class OrderInteractionTester extends
MockObjectTestCase {
  private static String TALISKER = "Talisker";

  public void testFillingRemovesInventoryIfInStock() {
    //setup - data
    Order order = new Order(TALISKER, 50);
    Mock warehouseMock = new Mock(Warehouse.class);

    //setup - expectations
    warehouseMock.expects(once()).method("hasInventory"
)
      .with(eq(TALISKER),eq(50))
      .will(returnValue(true));
    warehouseMock.expects(once()).method("remove")
      .with(eq(TALISKER), eq(50))
      .after("hasInventory");

    //exercise
    order.fill((Warehouse) warehouseMock.proxy());

    //verify
    warehouseMock.verify();
    assertTrue(order.isFilled());
  }

  public void
testFillingDoesNotRemoveIfNotEnoughInStock() {
    Order order = new Order(TALISKER, 51);
    Mock warehouse = mock(Warehouse.class);

    warehouse.expects(once()).method("hasInventory")
      .withAnyArguments()
      .will(returnValue(false));

    order.fill((Warehouse) warehouse.proxy());

    assertFalse(order.isFilled());
  }
```

Concentrez vous d'abord sur le test testFillingRemovesInventoryIfInStock , car j'ai pris quelques raccourcis avec l'autre test.

Tout d'abord, la phase d'initialisation est très différente. Pour commencer, elle est divisée en deux parties : les données et les attentes. La partie données initialise les objets qui nous intéressent, et en ce sens elle est similaire à l'initialisation traditionnelle. La différence réside dans les objets qui sont créés. le SCT est le même - une commande. Cependant le collaborateur n'est plus un entrepôt, mais un simulacre d'entrepôt - techniquement une instance de la classe Mock.

La deuxième partie de l'initialisation crée des attentes sur l'objet simulacre. Les attentes indiquent quelles méthodes devraient être appelées sur les simulacres quand le SCT est exécuté.

Une fois que les attentes sont en place, j'exécute le SCT. Après l'exécution je fais alors la vérification, qui a deux aspects. J'utilise des assertions sur le SCT - à peu près comme avant. Cependant je vérifie également les simulacres - en contrôlant qu'ils ont bien été appelés selon leurs attentes.

La différence clé ici est comment nous vérifions que la commande a fait ce qu'elle devait dans son interaction avec l'entrepôt. Avec la vérification d'état, nous faisons cela avec des assertions sur l'état de l'entrepôt. Les simulacres utilisent la vérification du comportement, et nous vérifions alors si la commande a fait les bons appels de méthodes sur l'entrepôt. Nous faisons cela pendant l'initialisation en disant au simulacre ce qu'il doit attendre, puis en demandant au simulacre de se vérifier lui-même durant la phase de vérification. Seule la commande est vérifiée à l'aide d'assertions, et si la méthode testée ne change pas l'état de la commande, alors il n'y a même pas d'assertions du tout.

Dans le deuxième test je fais plusieurs choses différemment. Premièrement je crée le simulacre d'une autre manière, en utilisant la méthode mock dans MockObjectTestCase au lieu du constructeur. C'est une méthode utilitaire de la bibliothèque jMock, qui me permet d'éviter de faire explicitement la vérification plus tard ; en effet tout simulacre crée avec cette utilitaire est automatiquement vérifié à la fin du test. J'aurais pu faire cela avec le premier test aussi, mais je voulais montrer la vérification d'une manière plus explicite pour bien montrer comment fonctionne le test avec des simulacres.

Deuxièmement, dans le second test, j'ai relâché les contraintes sur l'attente en utilisant withAnyArguments. La raison est que le premier test vérifie déjà que le nombre est bien passé à l'entrepôt, aussi le deuxième test n'a pas besoin de répéter cet élément de test. Si la logique de la commande doit être modifiée plus tard, alors un seul test échouera, ce qui facilitera l'effort de migration des tests. J'aurais également pu laisser entièrement de côté withAnyArguments, car c'est le fonctionnement par défaut.

4.Utilisation de EasyMock

Il y a un certain nombre de bibliothèques d'objets simulacres. Je rencontre assez fréquemment EasyMock, à la fois dans ses versions java et .NET. EasyMock permet également la vérification du comportement, mais a plusieurs différences de style avec jMock qui méritent d'être discutées. Voici à nouveau nos tests :

```
public class OrderEasyTester extends TestCase {
  private static String TALISKER = "Talisker";

  private MockControl warehouseControl;
  private Warehouse warehouseMock;

  public void setUp() {
    warehouseControl =
MockControl.createControl(Warehouse.class);
    warehouseMock = (Warehouse)
warehouseControl.getMock();
  }

  public void testFillingRemovesInventoryIfInStock() {
    //setup - data
    Order order = new Order(TALISKER, 50);

    //setup - expectations
    warehouseMock.hasInventory(TALISKER, 50);
    warehouseControl.setReturnValue(true);
    warehouseMock.remove(TALISKER, 50);
    warehouseControl.replay();

    //exercise
    order.fill(warehouseMock);

    //verify
    warehouseControl.verify();
    assertTrue(order.isFilled());
  }

  public void
testFillingDoesNotRemoveIfNotEnoughInStock() {
    Order order = new Order(TALISKER, 51);

    warehouseMock.hasInventory(TALISKER, 51);
    warehouseControl.setReturnValue(false);
    warehouseControl.replay();

    order.fill((Warehouse) warehouseMock);

    assertFalse(order.isFilled());
    warehouseControl.verify();
  }
}
```

EasyMock utilise la métaphore enregistrer/rejouer pour définir les attentes. Pour chaque objet collaborateur pour lequel vous désirez un simulacre, vous créez un objet de contrôle et un simulacre. Le simulacre implémente l'interface de l'objet collaborateur, tandis que l'objet de contrôle vous donne des fonctionnalités supplémentaires. Pour indiquer une attente, vous appelez la méthode sur le simulacre, avec les arguments que vous attendez. Vous faites ensuite un appel au contrôle si vous voulez une valeur de retour. Une fois que vous avez fini de définir les attentes, vous appelez replay sur le contrôle - à ce stade le simulacre termine l'enregistrement et est prêt à répondre au SCT. Une fois que c'est fait vous appelez verify sur le contrôle.

Il semble que les gens sont souvent troublés à la première vue de la métaphore enregistrer/rejouer, mais qu'ils s'y habituent rapidement. Elle a un avantage par rapport aux contraintes de jMock en ce que vous faites de vrais appels de méthodes sur le simulacre au lieu de spécifier des noms de méthodes dans des chaînes. Ce qui veut dire que vous pouvez utiliser la complétion de code de votre environnement de développement, et que tout remaniement de noms de méthodes mettra automatiquement à jour les tests. L'inconvénient est que vous ne pouvez pas relâcher les contraintes.

Les développeurs de jMock travaillent sur une nouvelle version qui permettra d'utiliser de vrais appels de méthodes.

5. La différence entre les simulacres et les bouchons

Quand ils ont été introduits pour la première fois, beaucoup de gens ont facilement confondu les objets simulacres avec la notion courante de test avec des bouchons. Depuis ils semblent que les différences sont mieux comprises (et j'espère que la précédente

version de cet article y a contribué). Cependant, pour comprendre complètement comment les simulacres sont utilisés, il est important de comprendre à la fois les simulacres et les autres types de doublures de tests ("doublures" ? Ne vous inquiétez pas si c'est un nouveau terme pour vous, attendez quelques paragraphes et tout deviendra clair).

Quand vous testez, vous vous focalisez sur un seul élément du logiciel à la fois - d'où le terme courant de test unitaire. Le problème est que pour faire fonctionner un élément particulier, vous avez souvent besoin d'autres éléments - par exemple dans notre exemple nous avons besoin de Warehouse.

Dans les deux types de tests que j'ai montré ci-dessus, le premier type utilise un vrai objet Warehouse, et le deuxième utilise un simulacre de Warehouse, lequel bien sûr n'est pas un vrai objet Warehouse. Utiliser un simulacre est donc un moyen de ne pas utiliser un vrai Warehouse dans le test, mais d'autres formes de "faux" objets sont également utilisées.

Le vocabulaire pour parler de ces notions devient vite confus - toutes sortes de mots sont utilisés : bouchon, simulacre, substitut, fantôme. Pour cet article je vais suivre le vocabulaire du livre de Gerard Meszaros. Il n'est pas utilisé par tout le monde, mais je pense que c'est un bon vocabulaire, et comme il s'agit de mon article j'ai le privilège de choisir quels mots utiliser.

Meszaros utilise le terme Doublure de Test comme terme générique pour tout objet utilisé à la place d'un vrai objet dans le but de tester. Ce terme correspond au cascadeur qui double un acteur dans un film (un des buts de Meszaros était d'éviter tout nom déjà largement utilisé). Meszaros définit alors quatre types particuliers de doublures:

- Fantômes : des objets que l'on fait circuler, mais qui ne sont jamais réellement utilisés. Habituellement ils servent juste à remplir des listes de paramètres.
- Substituts : des objets qui ont de véritables implémentations qui fonctionnent, mais qui généralement prennent des raccourcis qui les rendent impropre à l'utilisation en production (un bon exemple est une base de données en mémoire au lieu d'une vraie base de données).
- Bouchons : ces objets fournissent des réponses prédéfinies à des appels faits durant le test, mais généralement ne répondent à rien d'autre en dehors de ce qui leur a été programmé pour le test. Les bouchons peuvent aussi enregistrer de l'information concernant les appels, par exemple un bouchon de passerelle de courriels peut mémoriser les messages qu'il a "envoyé", ou encore seulement le nombre de messages qu'il a "envoyés".
- Simulacres : les objets dont nous parlons ici: des objets préprogrammés avec des attentes, lesquelles constituent une spécification des appels qu'ils s'attendent à recevoir.

Parmi toutes ces doublures, seuls les simulacres insistent sur la vérification du comportement. Les autres doublures peuvent (et généralement le font) utiliser la vérification d'état. En fait les simulacres se comportement vraiment comme les autres doublures dans la phase d'exécution, car ils ont besoin de faire croire au SCT qu'il parle à ses vrais collaborateurs - mais les simulacres diffèrent dans les phases d'initialisation et de vérification.

Pour explorer un peu plus les doublures de test, nous devons étendre notre exemple. Beaucoup de gens utilisent une doublure uniquement si le vrai objet est peu pratique à manipuler. Ici, disons que nous voulons envoyer un courriel si un Order échoue. Le problème est que nous ne pouvons pas envoyer de vrais courriels à des clients pendant nos tests. Donc à la place nous créons une doublure de notre système de courriels, que nous pouvons contrôler et manipuler.

Nous pouvons alors commencer à voir la différence entre les simulacres et les bouchons. Pour tester ce comportement d'envoi de courriels, nous pourrions écrire un simple bouchon comme ceci :

```
public interface MailService {
  public void send (Message msg);
}

public class MailServiceStub implements MailService {
  private List<Message> messages = new
ArrayList<Message>();
  public void send (Message msg) {
    messages.add(msg);
  }
  public int numberSent() {
    return messages.size();
  }
}
```

Nous pouvons alors utiliser la vérification d'état sur le bouchon comme suit :

```
class OrderStateTester...
  public void testOrderSendsMailIfUnfilled() {
    Order order = new Order(TALISKER, 51);
    MailServiceStub mailer = new MailServiceStub();
    order.setMailer(mailer);
    order.fill(warehouse);
    assertEquals(1, mailer.numberSent());
  }
```

Bien sûr c'est un test très simple - il vérifie seulement qu'un message a été envoyé. Nous n'avons pas testé qu'il a été envoyé à la bonne personne, ni qu'il avait le bon contenu, mais il suffira à illustrer mon propos.

En utilisant des simulacres, ce test serait bien différent :

```
class OrderInteractionTester...
  public void testOrderSendsMailIfUnfilled() {
    Order order = new Order(TALISKER, 51);
    Mock warehouse = mock(Warehouse.class);
    Mock mailer = mock(MailService.class);
    order.setMailer((MailService) mailer.proxy());

    mailer.expects(once()).method("send");
    warehouse.expects(once()).method("hasInventory")
      .withAnyArguments()
      .will(returnValue(false));

    order.fill((Warehouse) warehouse.proxy());
  }
}
```

Dans les deux cas j'utilise une doublure de test à la place du vrai service de courriel. La différence est dans le fait que le bouchon utilise la vérification d'état alors que le simulacre utilise la vérification du comportement.

Pour utiliser la vérification d'état sur le bouchon, j'ai besoin de quelques méthodes supplémentaires sur le bouchon pour faciliter la vérification. Par conséquent le bouchon implémente MailService mais ajoute des méthodes de test supplémentaires.

Les simulacres utilisent toujours la vérification du comportement, tandis qu'un bouchon peut faire les deux. Meszaros nomme les bouchons qui utilisent la vérification de comportement comme des Espions de Test. La différence réside dans la manière dont la doublure effectue exécution et vérification, et je vous laisse le soin d'explorer cela vous-même.

6. Les styles de test classique et orienté-simulacre

Maintenant j'en suis au point où je peux explorer la deuxième dichotomie entre le test classique et orienté-simulacre. Le point principal est quand utiliser un simulacre (ou une autre doublure).

Le **style classique de TDD** consiste à utiliser de vrais objets si possibles, et une doublure quand le vrai objet est trop compliqué à utiliser. Ainsi un adepte du TDD classique utiliserait le vrai Warehouse et un double pour le service de courriels. Le type exact de doublure ne compte pas tant que cela.

Un **adepte du style orienté-simulacre**, par contre, utilisera toujours un simulacre pour tout objet ayant un comportement intéressant, et donc dans ce cas pour la à la fois Warehouse et le service de courriels.

Bien que les différents outils de création de simulacres aient été conçus avec le style orienté-simulacre en tête, beaucoup d'adeptes du style classique les trouvent utiles pour créer des doubles.

Une conséquence importante du style orienté-simulacre est le Développement Dirigé par le Comportement (BDD) (Lien108). Le BDD a été initialement développé par mon collègue Dan North comme une technique pour aider les gens à mieux apprendre le Développement Dirigé par les Tests en insistant sur la technique de conception que représente le TDD. Cela a conduit à renommer les tests en comportements pour mieux explorer comment le TDD aide à réfléchir à ce qu'un objet doit faire. Le BDD suit une approche orientée-simulacre, mais il va plus loin, à la fois dans ses styles de nommage, et dans son désir d'intégrer la conception dans sa technique. Je ne vais pas plus loin ici, car le seul lien avec cet article est que le BDD est une autre variation du TDD qui tend à utiliser les simulacres. Je vous laisse suivre le lien pour plus d'informations.

7. Comment choisir parmi les différences ?

Dans cet article j'ai expliqué deux différences : vérification de l'état ou du comportement, style classique ou orienté simulacre. Quels sont les arguments à garder l'esprit quand on fait un choix entre ces styles ? Je vais commencer par le choix entre la vérification de l'état ou du comportement.

La première chose à considérer est le contexte. Pensons-nous à une collaboration facile, comme entre Order et Warehouse, ou une compliquée, comme entre Order et le service de courriels ?

Si c'est une collaboration facile alors le choix est simple. Si je suis un adepte du style classique, je n'utilise pas de simulacre, bouchon ou autre sorte de doublure. J'utilise le vrai objet et la vérification d'état. Si je suis un adepte du style orienté-simulacre, j'utilise un simulacre et la vérification du comportement. Pas de décision à prendre du tout.

Si c'est une collaboration compliquée, il n'y a pas de décision si je suis un adepte des simulacres - j'utilise juste des simulacres et la vérification du comportement. Si je suis un adepte du style classique, alors j'ai le choix, mais il n'est pas bien important. Habituellement les adeptes du style classique décident au cas par cas, en utilisant le chemin le plus rapide pour chaque situation.

Donc comme nous le voyons, la décision entre vérification de l'état ou du comportement n'est pas une grosse décision la plupart du temps. La vraie difficulté est de choisir entre TDD classique et orienté-simulacre. Il apparaît que les caractéristiques de la vérification de l'état et du comportement affectent cette discussion, et c'est là que je vais focaliser mon énergie.

Mais avant de le faire, je vais proposer un cas limite. De temps en temps vous rencontrez des choses dont il est vraiment difficile de vérifier l'état, même s'il ne s'agit pas de collaborations compliquées. Un bon exemple de cela est un cache. La particularité d'un cache est que vous ne pouvez pas dire à partir de son état comment le cache a fonctionné - c'est un cas où la vérification du comportement serait un choix sage même pour un adepte pur et dur du style TDD classique. Je suis sûr qu'il y a d'autres exceptions dans les deux directions.

Maintenant que nous approfondissons le choix entre styles classique/orienté-simulacre, il y a des tas de facteurs à considérer, donc je les ai organisés en groupes grossiers :

8. Le guidage du TDD

Les objets simulacres sont issus de la communauté XP, et l'une des caractéristiques principales de XP est son insistance sur le Développement Dirigé par les Tests - dans lequel la conception d'un système évolue par des itérations pilotées par l'écriture de tests.

Ainsi il n'y a rien de surprenant à ce que les adeptes du style orienté-simulacre parlent de l'effet sur la conception du test avec des simulacres. En particulier ils recommandent un style appelé Développement Dirigé par les Besoins. Avec ce style vous commencez à développer une histoire utilisateur en écrivant votre premier test pour l'extérieur de votre système, en faisant d'un certain objet interface votre SCT. En réfléchissant aux attentes sur les collaborateurs, vous explorez les interactions entre le SCT et ses voisins - et donc vous concevez effectivement l'interface externe du SCT.

Une fois que vous avez votre premier test qui tourne, les attentes sur les simulacres fournissent les spécifications pour la prochaine étape, et un point de départ pour les tests. Vous transformez chaque attente en un test sur un collaborateur et répétez le processus en progressant dans le système un SCT à la fois. Ce style est également nommé "extérieur-vers-intérieur", ce qui en est une très bonne description. Il fonctionne bien avec les systèmes organisés en couches. Vous commencez d'abord en programmant l'interface utilisateur en utilisant des simulacres des couches sous-jacentes. Ensuite vous écrivez vos tests pour la couche en-dessous, en parcourant graduellement le système une couche à la fois. C'est une approche très structurée et très contrôlée, dont beaucoup de gens pensent qu'elle est utile pour guider les novices dans la programmation orientée-objet et le TDD.

Le TDD classique ne fournit pas tout à fait le même guidage. Vous pouvez faire une approche progressive similaire, en utilisant des méthodes bouchons à la place de simulacres. Pour faire cela, chaque fois que vous avez besoin de quelque chose d'un collaborateur, vous codez simplement en dur la réponse nécessaire pour faire marcher le SCT. Une fois que vous avez la barre verte vous remplacez la réponse en dur par le code approprié.

Mais le TDD classique peut faire d'autres choses également. Un

style courant est "milieu-vers-extérieur". Dans ce style vous prenez une fonctionnalité et décidez ce dont vous avez besoin dans le domaine pour que cette fonctionnalité marche. Vous faites faire aux objets du domaine ce dont vous avez besoin, et une fois qu'ils marchent vous établissez l'interface utilisateur au-dessus. En faisant cela vous pouvez très bien ne rien avoir à doubler du tout. Beaucoup de gens aiment cette approche car elle concentre l'attention sur le modèle du domaine d'abord, ce qui empêche la logique du domaine de contaminer l'interface utilisateur.

Je voudrais souligner que les adeptes des styles orienté-simulacre et classique font cela une seule histoire utilisateur à la fois. Il y a une école de pensée qui construit les applications couche par couche, ne commençant pas une nouvelle couche tant que celle en cours n'est pas terminée. Au contraire les adeptes du style classique et des simulacres tendent à avoir une approche agile et préfèrent des itérations de petite taille. Par conséquent ils travaillent fonctionnalité par fonctionnalité plutôt que couche par couche.

9. L'initialisation des classes de test

Avec le TDD classique vous devez créer non pas seulement le SCT mais aussi tous les collaborateurs dont le SCT à besoin pour répondre au test. Bien que l'exemple ci-dessus n'ait que peu d'objets, les tests réels impliquent souvent un grand nombre de collaborateurs. Habituellement ces objets sont créés et supprimés lors de chaque exécution des tests.

Les tests avec simulacres, cependant, ont seulement besoin de créer le SCT et des simulacres pour ses collaborateurs immédiats. Ceci peut éviter un peu du travail impliqué dans la construction de classes de test complexes (au moins en théorie. J'ai entendu des histoires d'initialisations de simulacres pas mal complexes, mais cela était peu être dû à une mauvaise utilisation des outils).

En pratique les testeurs classiques tendent à réutiliser autant que possible les initialisations de test complexes. La manière la plus simple est de mettre le code d'initialisation dans une méthode setup xUnit. Des initialisations plus compliquées peuvent avoir besoin d'être utilisées par plusieurs classes de test, alors dans ce cas vous créez des classes spéciales pour générer les initialisations. Je les appelle habituellement des Mères d'objets (Lien109), en me basant sur une convention de nommage utilisée dans un projet XP précoce chez ThoughtWorks. Utiliser des mères est essentiel dans le test classique de grande envergure, mais les mères représentent du code supplémentaire qui a besoin d'être maintenu, et tout changement au niveau des mères peut avoir des répercussions en chaîne à travers les tests. Il peut aussi y avoir une pénalité de performance à l'initialisation - bien que je n'ai pas entendu dire que ce soit un problème sérieux si cela est fait correctement. La création de la plupart des objets d'initialisation est bon marché, et quand ce n'est pas le cas ils sont habituellement doublés.

En conséquence j'ai entendu chaque style accuser l'autre de représenter trop de travail. Les partisans des simulacres disent que créer les initialisations est un gros effort, mais les classiques disent qu'elles sont réutilisables alors qu'il faut créer des simulacres pour chaque test.

10. L'isolation des tests

Si vous introduisez un bug dans un système avec du test basé sur des simulacres, généralement il fera échouer uniquement les tests dont le SCT contient le bug. Avec l'approche classique, cependant, n'importe quel test d'objet client peut aussi échouer, ce qui conduit à des échecs aux endroits où l'objet fautif est utilisé comme collaborateur dans le test d'un autre objet. Par conséquent, un échec dans un objet très utilisé cause une cascade d'échecs de test à travers le système.

Les testeurs utilisant des simulacres considèrent cela comme un problème majeur; il conduit à beaucoup de débogage pour trouver la cause racine de l'erreur et la corriger. Cependant les classiques n'expriment pas cela comme une source de problèmes. Habituellement le coupable est assez facile à identifier en examinant quels tests échouent et les développeurs peuvent alors dire quels échecs dérivent de la cause racine. De plus si vous testez régulièrement (comme vous le devriez) alors vous savez que l'échec a été causé par ce que vous avez édité en dernier, donc il n'est pas difficile de trouver l'erreur.

La granularité des tests peut être un facteur significatif ici. Puisque les tests classiques mettent en jeu de nombreux objets réels, vous trouvez souvent un test particulier qui est le test primaire pour un groupe d'objets, plutôt que pour un seul objet. Si ce groupe comprend beaucoup d'objets, alors trouver la vraie cause d'un bug peut être beaucoup plus difficile. Ce qui arrive ici c'est ce que les tests ont une granularité trop grossière.

Il est probable que les tests avec des simulacres sont moins enclins à souffrir de ce problème, car la convention est de faire des simulacres pour tous les objets au-delà de l'objet primaire, ce qui rend clair que des tests de granularité plus fine sont nécessaires pour les collaborateurs. Ceci étant dit, il est aussi vrai qu'utiliser des tests à granularité trop grossière n'est pas nécessairement un échec du test classique en tant que technique, c'est plutôt dû à une mauvaise application du test classique. Une bonne règle empirique est de s'assurer d'isoler des tests de fine granularité pour chaque classe. Alors que des groupes sont parfois raisonnables, ils devraient être limités à un tout petit nombre d'objets - pas plus qu'une douzaine. De plus, si vous vous trouvez face à un problème de débogage dû à des tests de granularité trop grossière, vous devriez déboguer dans une optique TDD, en créant des tests de plus fine granularité au fur et à mesure que vous avancez.

En l'essence les tests classiques xunit ne sont pas juste des tests unitaires, mais aussi de mini tests d'intégration. Par conséquent beaucoup de gens aiment le fait que des tests sur des clients peuvent attraper des erreurs que les tests principaux pour un objet auraient ratées, en sondant particulièrement les zones où les classes interagissent. Les tests avec les simulacres perdent cette qualité. De plus vous courrez aussi le risque que les attentes sur les simulacres soient incorrectes, ce qui résulte en des tests unitaires qui passent au vert mais masquent des erreurs inhérentes.

A ce stade je dois souligner que quelque soit le style de test que vous utilisez, vous devez le combiner avec des tests d'acceptance de granularité plus grossière, tests qui opèrent à travers tout l'ensemble du système. J'ai souvent croisé des projets qui étaient en retard dans l'utilisation de tests d'acceptance, et qui l'ont regretté.

11. Le couplage des tests avec les implémentations

Quand vous écrivez un test avec simulacres, vous testez les appels du SCT vers l'extérieur pour vous assurer qu'il parle correctement à ses fournisseurs. Un test classique s'intéresse uniquement à l'état final - pas à la manière dont cet état a été obtenu. Les tests avec simulacres sont ainsi plus fortement couplés à l'implémentation d'une méthode. Changer la nature des appels aux collaborateurs cassera généralement un test avec simulacre.

Ce couplage entraîne plusieurs préoccupations. La plus importante est l'effet sur le TDD. Dans le cas des tests avec simulacres, écrire le test vous fait réfléchir à l'implémentation du comportement - et en effet les testeurs avec simulacres voient cela comme un avantage. Les classiques, cependant, pensent qu'il est important de réfléchir seulement à ce qui arrive de l'interface externe, et de laisser de côté toute considération d'implémentation jusqu'à ce que vous ayez fini d'écrire le test.

Le couplage avec l'implémentation interfère également avec le remaniement, puisque les changements d'implémentation risquent beaucoup plus de casser les tests que dans le cas du test classique.

Cela peut être aggravé par la nature même des boites à outils de simulacres. Souvent les outils de simulacres spécifient des appels de méthodes et des appariements de paramètres très spécifiques, même quand ils ne sont pas pertinents pour le test en question. Un des buts de la boite à outils jMock est d'être plus flexible dans sa spécification des attentes, pour autoriser le relâchement des attentes dans les zones où elles n'ont pas d'importance, au prix de l'utilisation de chaînes qui rendent les remaniements plus délicats.

12. Le style de conception

Pour moi, l'un des aspects les plus fascinants de ces styles de test est la manière dont ils influencent les décisions de conception. Comme j'ai parlé avec les deux types de testeurs, je suis devenu conscient de quelques différences entre les conceptions encouragées par les styles, mais je suis certain que je ne fais qu'égratigner la surface de ce sujet.

J'ai déjà mentionné une différence dans la manière d'aborder les couches. Les tests avec simulacres supportent une approche "extérieur-vers-intérieur" tandis que les développeurs qui préfèrent travailler à partir du modèle du domaine tendent à préférer le test classique.

A un niveau moins élevé, j'ai noté que les testeurs orientés-simulacres tendent à s'éloigner des méthodes qui retournent des valeurs, pour favoriser les méthodes qui agissent sur un objet collecteur. Prenez l'exemple du comportement consistant à rassembler de l'information d'un groupe d'objets pour créer un rapport sous forme de chaîne de caractères. Une manière courante de procéder est d'avoir une méthode de rapport qui appelle sur les différents objets des méthodes retournant des chaînes, et qui assemble la chaîne résultat dans une variable temporaire. Un testeur orienté-simulacre passera probablement un tampon de caractères aux différents objets et leur fera ajouter les différentes chaînes à ce tampon - traitant ainsi le tampon comme un paramètre collecteur.

Les testeurs avec simulacres parlent également plus d'éviter les désastres en chaînes - les chaînes de méthodes du type getThis().getThat().getTheOther(). Éviter les chaînes de méthodes est également connu sous le nom de la Loi de Demeter. Alors que les chaînes de méthodes sont une mauvaise odeur, le problème opposé des objets intermédiaires boursouflés de méthodes de transfert est également une mauvaise odeur. (J'ai toujours senti que je serais plus confortable avec la Loi de Demeter si elle était appelée la Suggestion de Demeter).

L'une des choses les plus difficiles à comprendre dans la conception orientée-objet est le principe "Fais au lieu de demander" (Lien110) qui vous encourage à dire à un objet de faire quelque chose, au lieu de lui extraire des données pour faire la chose en question dans du code client. Les testeurs orienté-simulacres disent que le test avec simulacres favorise ce principe et évite les confettis de code "get..." qui se répand dans beaucoup trop de code ces temps-ci. Les classiques avancent qu'il y a beaucoup d'autres manières de faire cela.

Un problème reconnu avec la vérification d'état est qu'elle peut conduire à la création de méthodes requêtes servant seulement à supporter la vérification. Il n'est jamais confortable d'ajouter des méthodes à l'API d'un objet dans le seul but de tester; utiliser la vérification du comportement évite ce problème. Le contre-argument est que de telles modifications sont souvent mineures en pratique.

Les testeurs orientés-simulacres favorisent les interfaces de rôles (Lien111) et assurent qu'utiliser ce style de test encourage la création de plus d'interfaces de rôles, puisque chaque collaboration a son propre simulacre et est donc plus sujette à devenir une interface de rôle. Ainsi dans mon exemple ci-dessus concernant la génération d'un rapport en utilisant un tampon de caractères, un testeur orienté-simulacre serait plus enclin à inventer un rôle particulier qui aurait du sens dans ce domaine, et qui pourrait être implémenté par un tampon de chaînes.

Il est important de se rappeler que cette différence de style de conception est un élément de motivation clé pour la plupart des testeurs orientés-simulacres. Les origines du TDD étaient un désir d'obtenir de solides tests automatisés de régression qui supporteraient une conception évolutionnaire. Sur le chemin ses adeptes ont découvert qu'écrire des tests d'abord représentait une amélioration significative du processus de conception. Les adeptes du style orienté-simulacre ont une forte idée de quelle conception est une bonne conception, et ont développé des bibliothèques de simulacres principalement pour aider les gens à développer ce style de conception.

13.Faut-il être un testeur classique ou orienté-simulacres ?

Je trouve qu'il est difficile de répondre avec certitude. Personnellement j'ai toujours été un adepte du bon vieux TDD classique et jusque-là je ne vois pas de raison de changer. Je ne vois aucun bénéfice irréfutable pour le TDD avec simulacres, et je suis préoccupé par les conséquences de coupler les tests avec l'implémentation.

Ceci me frappe particulièrement quand j'observe un programmeur orienté-simulacre. J'aime vraiment le fait que pendant que vous écrivez le test, vous vous concentrez sur le résultat du comportement, pas sur comment il est fait. Un adepte des simulacres réfléchit constamment à l'implémentation du SCT pour pouvoir écrire les attentes. Cela me paraît vraiment anti-naturel.

Je souffre également de l'inconvénient de ne pas essayer le TDD avec simulacres sur autre choses que des applications jouets. Comme je l'ai appris du TDD lui-même, il est souvent difficile de juger une technique sans l'essayer sérieusement. Je connais beaucoup de bons développeurs qui sont des adeptes convaincus et très heureux des simulacres. Donc bien que je sois toujours un classique convaincu, j'ai préféré présenter les deux arguments aussi équitablement que possible de telle sorte que vous puissiez vous faire votre propre idée.

Donc si le test avec simulacres vous paraît attractif, je vous suggère de l'essayer. Cela en vaut particulièrement la peine si vous avez des problèmes dans certaines zones que le test avec simulacre est conçu pour améliorer. Je vois deux zones principales ici. La première si vous passez beaucoup de temps à déboguer quand des tests échouent, parce qu'ils n'échouent pas proprement et ne vous disent pas où est le problème. La deuxième

si vos objets ne contiennent pas suffisamment de comportements ; le test avec simulacres peut encourager l'équipe de développement à créer des objets plus riches en comportements.

14. Considérations finales

Au fur et à mesure que grandit l'intérêt pour les outils xUnit et le Développement Dirigé par les Tests, de plus en plus de gens croisent les simulacres. Le plus souvent, ils apprennent une partie des outils pour simulacres, sans comprendre complètement la division classique/orienté-simulacre sur lesquels ils reposent. Quelque soit votre côté dans cette division, je pense qu'il est utile de comprendre cette différence de vue. Alors que vous n'avez pas besoin d'être orienté-simulacres pour trouver pratiques les outils de simulacres, il est utile de comprendre le raisonnement qui guide la plupart des décisions de conception du logiciel.

Le but de cet article était, et est toujours, de souligner ces différences et d'exposer les compromis entre elles. Il y a plus dans la réflexion orientée-simulacres que ce que j'ai eu le temps d'approfondir, en particulier ses conséquences sur le style de conception. J'espère que dans les toutes prochaines années nous verrons plus de choses écrites sur cela, et que cela approfondira notre compréhension des conséquences fascinantes de l'écriture de tests avant le code.

Retrouvez la traduction par Bruno Orsier de l'article de Martin Fowler en ligne : Lien112

SharePoint

Les derniers tutoriels et articles

Développer un event handler pour WSS 3/MOSS 2007

Ce tutoriel illustre comment créer un event handler (gestionnaire de procédures évènementielles attachées à une liste) dans WSS 3/MOSS 2007

1. Introduction

Un event handler est une librairie que l'on déploie sur le serveur Sharepoint et que l'on attache à une liste pour déclencher une ou plusieurs actions lorsqu'un élément de cette liste est ajouté/modifié/supprimé. Il y a une multitude de cas d'utilisation possible. Je dirais qu'on peut les utiliser pour presque tout sauf pour des actions nécessitant une IHM et donc une intervention humaine quelconque (approbation de document par ex). Pour ce cas précis, on préfèrera développer un workflow.

Pour ce tutoriel, nous allons développer un Event Handler qui contrôlera la taille du document que nous chargerons dans la liste et qui annulera l'insertion en cas de dépassement d'une certaine limite. Par ailleurs, et pour gérer plus qu'un seul évènement, il ajoutera un élément d'audit dans une liste prévue à cet effet à chaque fois qu'un élément de la libraire de document est supprimé. Il ajoutera également une annonce dans la liste d'annonces prévue à cet effet lorsqu'un document aura été ajouté avec succès dans la librairie.

2. Développement

Il existe deux types d'évènement dans les event handlers. Les évènements synchrones et asynchrones. Les évènements synchrones nous permettent d'intervenir lors d'opérations en cours comme l'addition, la suppression, ou la modification d'un élément et nous permettent le cas échéant d'interrompre le processus.

Les évènements asynchrones interviennent lorsque l'opération sur laquelle on désire intervenir est terminée. Ils ne peuvent donc pas servir à interrompre une opération.

2.1. Les évènements intervenant sur des éléments de liste

Tous les évènements ci-dessous sont des membres de **SPItemEventReceiver**

2.1.1. Evènements synchrones

ItemAdding
Intervient lorsqu'un nouvel élément est ajouté dans une liste
ItemAttachmentAdding
Intervient avant l'addition d'une pièce jointe.
ItemAttachmentDeleting
Intervient avant la suppression d'une pièce jointe
ItemCheckingIn
Intervient avant le check-in d'un document. Le check-in consiste à valider une modification préalablement effectuée.
ItemCheckingOut
Intervient avant la mise en check-out d'un fichier. Le check-out consiste à s'accaparer une copie d'un fichier que l'on va pouvoir modifier. Celui-ci sera en état brouillon tant qu'il n'aura pas été validé par un check-in
ItemDeleting
Intervient avant la suppression d'un document.
ItemFileMoving
Intervient avant le déplacement d'un document.
ItemUnCheckingOut
Intervient avant l'annulation d'un check-out de document.
ItemUpdating
Intervient avant la mise à jour d'un élément.

2.1.2. Evènements asynchrones

ItemAdded
Intervient après l'ajout d'un élément.

ItemAttachementAdded
Intervient après l'addition d'une pièce jointe par upload ou par ajout direct.

ItemAttachementDeleted
Intervient après la suppression d'une pièce jointe.

ItemCheckedIn
Intervient après le check-in d'un document.

ItemCheckedOut
Intervient après le check-out d'un document.

ItemDeleted
Intervient après la suppression d'un élément.

ItemFileMoved
Intervient après le déplacement d'une pièce jointe.

ItemUncheckedOut
Intervient après l'annulation d'un document en check-out.

ItemUpdated
Intervient après la modification d'un élément.

2.2. Séquençage des évènements

Même si cela paraît évident, le séquençage des évènements ne se produit pas toujours forcément comme on pourrait le croire. Par exemple, lorsque l'on ajoute un nouveau document dans une librairie de documents, les évènements ItemUpdating et ItemUpdated se produisent alors qu'on aurait pu croire que seuls les évènements ItemAdding et ItemAdded seraient déclenchés. Il est donc nécessaire de bien comprendre la séquence d'évènements afin d'éviter toute collision. Vous trouverez un petit programme que j'ai réalisé dans la section téléchargement qui crée un fichier log dans c:\temp au format HTML et qui ajoute une ligne dans ce fichier dès qu'un évènement est déclenché. Il vous sera alors facile de comprendre le séquençage.

J'ai aussi créé une petite application console qui permet d'associer tous les évènements de l'assemblage pour une liste donnée. Vous n'aurez qu'à lire le fichier lisezmoi.txt pour voir comment l'utiliser (très simple)

2.3. Création du projet et premiers pas

A l'heure où j'écris ce tutoriel, il n'existe pas de template visual studio pour les event handlers mais rassurez-vous, c'est très simple. Voici les étapes basiques pour démarrer

- Créez un nouveau projet de type "Class Library"
- Ajoutez la référence Microsoft.Sharepoint.dll qui se trouve généralement dans c:\Program Files\Common Files\Microsoft Shared\web server extensions\12\isapi
- Ajoutez la directive using Microsoft.Sharepoint
- Faites dériver votre classe de SPItemEventReceiver

Après ces étapes basiques, vous êtes réellement prêt à démarrer l'écriture de votre event handler.

2.4. Propriétés intéressantes

BeforeProperties
Expose les colonnes internes telles que title, filesize et les colonnes personnelles avec leur ancienne valeur. On peut par exemple comparer les valeurs de cette collection avec celles de AfterProperties pour vérifier si tel ou tel champ a été modifié. Cette collection est en lecture seule.

AfterProperties
Expose les colonnes internes telles que title, filesize et les colonnes personnelles avec leur nouvelle valeur. Cette collection est accessible en lecture-écriture. On peut donc l'utiliser pour attribuer des valeurs à certains champs. Si vous désirez créer un champ calculé personnel par exemple qui dépasse les limites d'un champ calculé que Sharepoint permet de faire.

ListItem
Expose toutes les colonnes de la liste. Est accessible en lecture-écriture et n'est disponible que pour les évènements asynchrones

WebUrl
Expose l'url du site contenant la liste qui déclenche l'évènement

BeforeUrl et AfterUrl
Contiennent respectivement l'url de l'item avant et après l'exécution de l'évènement. Lors d'une suppression par exemple, AfterUrl sera vide

Cancel
Mise à True, cette propriété permet d'avorter une opération. Elle n'est disponible que pour les évènements synchrones.

ErrorMessage
Permet d'afficher un message d'erreur personnel à l'utilisateur dans Sharepoint.

UserLoginName et UserDisplayName
Contiennent respectivement le nom de l'utilisateur et son login

2.5. La gestion d'erreurs

Un event handler n'est autre qu'une DLL classique mais celle-ci sera exécutée par Sharepoint. Toute erreur provoquée par votre code ne stoppera pas une opération en cours. Donc, si par exemple, un nouvel élément est ajouté à une liste et que vous avez défini un event handler sur l'ajout d'élément qui provoque une erreur, l'élément sera tout de même ajouté à la liste et un Event Log sera automatiquement créé par Sharepoint pour rapporter votre erreur

Vous pouvez néanmoins utiliser les try/catch/finally comme pour n'importe quelle autre DLL. Simplement, si vous ne contrôlez pas les exceptions pouvant survenir dans votre DLL, la couche supérieure de Sharepoint fera un catch et loggera l'évènement dans l'event viewer

2.6. Déboguer un event handler

Si vous êtes sur le serveur, vous pouvez directement déboguer à l'aide de visual studio. Vous devez ouvrir votre event handler, placer un point d'arrêt (par ex dans le constructeur) et attacher Visual Studio à l'un des processus w3p. Pour ce faire, procédez comme suit:

Si vous pouvez faire ce que vous voulez sur le serveur, je vous conseille d'arrêter tous les pools d'application dans IIS excepté celui qui exécute votre application Sharepoint
Ensuite, ouvrez votre projet dans visual studio
Cliquez sur Debug->Attach to process
Localisez le process W3p (si vous avez fait l'étape 1, vous ne devriez en avoir que un ou deux)

Cliquez sur Attach
Allez dans l'interface de Sharepoint et provoquez une action supposée déclencher votre event. Normalement, Visual Studio devrait clignoter et vous permettre de déboguer pas à pas.

2.7. Eviter de se mordre la queue

Si vous écrivez un event handler qui se déclenche sur la modification d'un élément et que dans votre event, vous modifiez à votre tour l'item courant (sa sécurit par exemple), vous allez involontairement redéclencher votre event puisque vous aurez procédé à une mise à jour. Pour éviter de se mordre la queue, Sharepoint permet de désactiver le déclenchement des event handlers et de les réactiver.

Pour les désactiver, vous pouvez utiliser DisableEventFiring (Lien113) et pour les réactiver vous pouvez utiliser EnableEventFiring (Lien114). Sachez néanmoins que la réactivation des events est automatiquement rétablie lorsque votre event se termine.

2.8. Notre code

Tous les commentaires intéressants se trouvent dans le code.

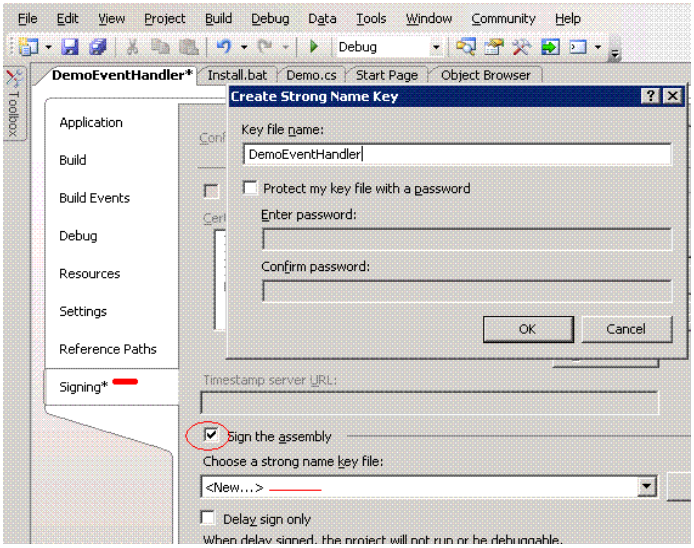
```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
using Microsoft.SharePoint;

namespace DemoEventHandler
{
    public class Demo : SPItemEventReceiver
    {
        const int MaxFileSize = 5000; //Taille maximale
de 5kb



        /// <summary>
        /// Cet évènement est déclenché lorsque le
document est en passe d'être ajouté à la liste
        /// </summary>
        /// <param name="properties"></param>
        public override void
ItemAdding(SPItemEventProperties properties)
        {
            //Capture de la taille du fichier uploadé
            int ItemFileSize =
Convert.ToInt16(properties.AfterProperties["vti_filesiz
e"].ToString());
            //Si elle dépasse la taille maximale
autorisée
            if (ItemFileSize > MaxFileSize)
            {
                //On affiche un message d'erreur
                properties.ErrorMessage = "Le fichier
que vous tentez de charger dans la liste est trop gros,
max " +
                    MaxFileSize.ToString();
                //On annule l'opération, l'élément ne
sera donc pas ajouté.
                properties.Cancel = true;
            }


        }
        /// <summary>
        /// Cet évènement est déclenché lorsque le
document a été ajouté à la liste.
        /// </summary>
        /// <param name="properties"></param>
        public override void
ItemAdded(SPItemEventProperties properties)
        {
            //Construction de l'annonce en récupérant
les propriétés Name et l'URL du document uploadé.
            StringBuilder MessageBody = new
StringBuilder();
            MessageBody.Append("Le document ");
            MessageBody.Append(properties.ListItem["Nam
e"]);
            MessageBody.Append(" créé par ");
            MessageBody.Append(properties.ListItem["Cre
ated By"]);
            MessageBody.Append(" est disponible <a
href='");
            MessageBody.Append(properties.ListItem["Enc
odedAbsUrl"]);
            MessageBody.Append("'>ici</a>");
            AddToList("Announcements",
MessageBody,properties.WebUrl,"Body","Nouveau
document!");
        }


        /// <summary>
        /// Cet évènement est déclenché lorsqu'un
document est supprimé
        /// </summary>
        /// <param name="properties"></param>
        public override void
ItemDeleted(SPItemEventProperties properties)
        {
            //Ajout d'un élément dans la table
AuditDocs qui dit quel document a été supprimé et par
qui
            StringBuilder MessageBody = new
StringBuilder();
            MessageBody.Append("L'élément ");
            MessageBody.Append(properties.ListItemId);
            MessageBody.Append(" ");
            MessageBody.Append(properties.BeforeUrl);
            MessageBody.Append(" a été supprimé par ");
            MessageBody.Append(properties.UserDisplayNa
me);
            AddToList("AuditDocs",MessageBody,propertie
s.WebUrl,"Title","");
        }
        /// <summary>
        /// Ajout d'un élément dans la liste "ListName"
        /// </summary>
        /// <param name="ListName"></param>
        /// <param name="Message"></param>
        /// <param name="Url"></param>
        /// <param name="FieldName"></param>
        /// <param name="Title"></param>
        private void AddToList(string
ListName,StringBuilder Message, string Url,string
FieldName,string Title)
        {
            SPSite Site = null;
            SPWeb Web = null;
            try
            {
                Site = new SPSite(Url);
                Web = Site.OpenWeb();
                SPList AuditList = Web.Lists[ListName];

                SPListItem AuditItem =
AuditList.Items.Add();
                if (Title != String.Empty)
                {
                    AuditItem["Title"] = Title;
                }
                AuditItem[FieldName] =
Message.ToString();
                AuditItem.Update();
```

```
            }
            finally
            {
                if (Web != null)
                    Web.Close();
                if (Site != null)
                    Site.Close();
            }
        }
    }
}
```

Pour rendre le code ci-dessus utilisable, il faut compiler le projet de type "Class Library" et ensuite le déployer dans la GAC. Les étapes de déploiement sont expliquées ci-dessous.

2.9. Signature de l'assemblage

La première chose à faire pour pouvoir déployer son assemblage dans la GAC (global assembly cache) est de le signer. Pour cela, sélectionnez votre projet dans l'explorateur de solution de Visual Studio et cliquez sur "Properties", ensuite sur l'onglet "Signing". Vous devriez obtenir ceci



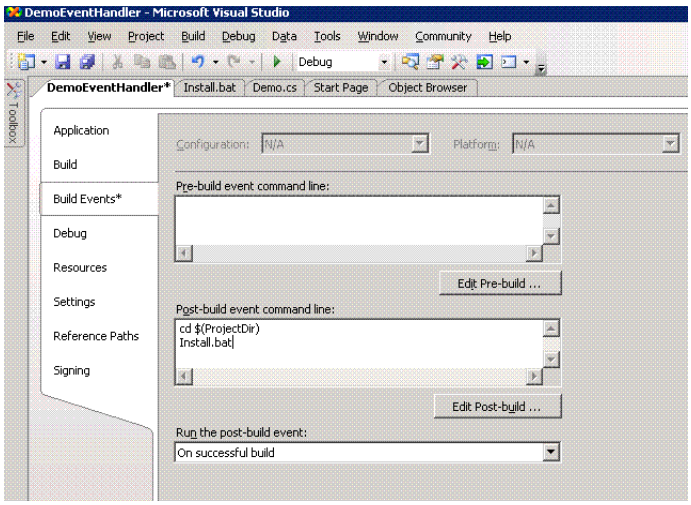
2.10. Automatiser l'exécution du fichier bat

L'utilisation d'un fichier bat facilite grandement le déploiement automatique de notre DLL dans la GAC. Il suffit donc d'ajouter un fichier à notre projet contenant les lignes suivantes:

```
"%programfiles%\Microsoft Visual Studio 8\SDK\v2.0\Bin\gacutil.exe" -uf DemoEventHandler
"%programfiles%\Microsoft Visual Studio 8\SDK\v2.0\Bin\gacutil.exe" -if bin\Debug\DemoEventHandler.dll
iisreset
```

La première ligne désinstalle notre DLL de la GAC, la deuxième le réinstalle et la troisième redémarre IIS. Notez qu'il peut être préférable de redémarrer uniquement le pool d'application comme expliqué ici

Pour automatiser l'exécution du fichier bat, allez dans les propriétés de votre projet et remplissez le post-build event comme illustré ci-dessous.



Après cette opération, votre event handler sera automatiquement déployé dans la GAC dès que vous compilerez votre projet.

3. Associer l'event handler à une liste

L'API Sharepoint nous donne la possibilité d'associer notre assemblage à une ou plusieurs listes. Il n'y a en effet aucune limite en matière d'association. Vous pouvez dès lors associer le même assemblage à trois listes différentes par exemple.

Depuis WSS 3 et MOSS 2007, il n'y a plus d'interface prévue dans la centrale d'administration pour associer un assemblage à une liste. Voici comment effectuer l'association

3.1. Associer par le code

```
SPSite Site = new SPSite("urldusite");
SPWeb Web = Site.OpenWeb();
Web.Lists["LaListeCible"].EventReceivers.Add(SPEventReceiverType.<Type d'évènement>,Signature de l'assemblage,Nom de la classe);
```

Ce qui donne dans un cas concret

```
string AssemblySignature="DemoAssembly, Version=1.0.0.0, Culture=neutral, PublicKeyToken=74cdd0bb8f510c15";
string ClassName="DemoClass";
Web.Lists["Shared Documents"].EventReceivers.Add(SPEventReceiverType.ItemAdding,AssemblySignature,ClassName);
Web.Lists["Shared Documents"].EventReceivers.Add(SPEventReceiverType.ItemAdded,AssemblySignature,ClassName);
```

Vous noterez que si vous implémentez plusieurs évènements dans un seul et même assembly, vous devrez néanmoins associer les évènements un par un à la liste.

3.2. Associer à l'aide d'un outil tiers

Etant donné que Microsoft n'a pas délivré d'outil spécifique pour gérer les event handlers mais offre une API permettant de les gérer, quelques sociétés ont déjà développé des DemoWare, Shareware etc...Parmi eux, il y a l'event handler explorer (Lien115). Il ne fait rien d'autre que ce qui est montré ci-dessus mais c'est plus pratique.

3.3 Visualiser les évènements existants

Si vous désirez connaître les évènements déjà présents pour une liste donnée, vous pouvez soit utiliser l'outil dont je viens de parler, soit l'API comme ceci:

```
SPSite Site = new SPSite("urldusite");
SPWeb Web = Site.OpenWeb();
```

```
foreach (SPEventReceiverDefinition def in
Web.Lists["Shared Documents"].EventReceivers)
{
    Console.WriteLine("Type {0} Assemblage lié
{1}",def.Type,def.Assembly);
}
```

Retrouvez l'article de Stéphane Eyskens en ligne : Lien117

La FAQ Sharepoint

De quel système d'exploitation ai-je besoin pour fonctionner avec Sharepoint?

Vous devez disposer de Windows 2003 serveur tant pour le serveur lui-même qu'en tant que plateforme de développement.

Sharepoint, c'est quoi au juste?

Sharepoint est un produit permettant de mettre au point des environnements intranet/extranet riches de manière efficace et rapide.

Il est encore principalement orienté gestion documentaire mais permet également de réaliser tous type d'application, surtout dans sa version 2007

Sur quelles technologies se base Sharepoint?

Sharepoint repose sur une architecture web composée de IIS et ASP.NET 2.0. Il utilise aussi le CLR 3.0 dont WF est issu. Tout développement personnel consiste en général en un composant ASP.NET que l'on incorpore dans l'architecture Sharepoint.

Est-ce que WSS 3.0 est gratuit?

Oui. WSS est un composant téléchargeable gratuitement sur le site de Microsoft. Cependant, étant donné que WSS ne s'installe que sur un serveur Windows 2K3, il vous faut disposer d'une CAL (Client Access Licence) pour ce serveur.

Principales différences entre WSS et MOSS (liste non exhaustive)?

- WSS est gratuit alors que MOSS est payant
- WSS n'englobe "que" les fonctionnalités de base de Sharepoint. MOSS est beaucoup plus riche en fonctionnalités "out of the box"
- MOSS intègre des services tels que Forms Services, Excel Services, le Business Data Catalog pas WSS
- MOSS dispose d'un moteur de recherche beaucoup plus poussé
- MOSS dispose d'une galerie de WebParts beaucoup plus complète que celle de WSS

Quel type de projet doit-on s'attendre à effectuer avec Sharepoint?

Tout projet nécessitant de la gestion documentaire sera grandement facilité avec Sharepoint. Tout projet nécessitant des processus d'approbation (workflow) également.

Sharepoint offre un cadre de sécurité (authorization) basé sur des rôles. Il est ensuite assez facile d'attribuer des rôles/permissions particulières sur les objets (listes, bdc…) contenus dans nos applications.

On peut techniquement réaliser tous type d'application Web car Sharepoint repose sur ASP.NET pour autant que l'on comprenne la philosophie du produit et que l'on essaye d'intégrer au mieux nos développements dans le canevas Sharepoint.

Il est évident qu'il faut essayer d'utiliser au maximum les fonctionnalités intégrées et ne pas vouloir réinventer la roue à chaque fois. Cependant, dans de nombreux cas, il est nécessaire de développer des composants/processus personnels et Sharepoint ouvre entièrement la porte aux développeurs.

Retrouvez ces questions et de nombreuses autres sur la FAQ Sharepoint : Lien118

Liens

Lien1 : http://www.opensymphony.com/sitemesh/
Lien2 : http://www.onjava.com/pub/a/onjava/2004/09/22/sitemesh.html
Lien3 : http://www.opensymphony.com/sitemesh/faq.html
Lien4 : http://loicmathieu.free.fr/wordpress/index.php?p=40
Lien5 : http://pcaboche.developpez.com/article/design-patterns/programmation-modulaire/?page=page_2#L1.2
Lien6 : http://loic-mathieu.developpez.com/java/tutoriel/sitemesh-intro/
Lien7 : http://java.sun.com/javase/6/docs/technotes/guides/jni/index.html
Lien8 : https://jna.dev.java.net/
Lien9 : http://blog.developpez.com/index.php?blog=51&p=4221&more=1&c=1&tb=1&pb=1
Lien10 : http://java.developpez.com/faq/java/
Lien11 : http://sourceforge.net/project/showfiles.php?group_id=117793
Lien12 : http://www.dsl.uow.edu.au/~sk33/php5java.htm
Lien13 : http://php-java-bridge.sourceforge.net/pjb
Lien14 : http://www.rjohnson.id.au/wordpress/2007/02/04/jasper-reports-and-php/
Lien15 : http://jasperforge.org/sf/projects/jasperreports
Lien16 : http://dev2dev.bea.com/pub/a/2007/02/php-java-bridge.html
Lien17 : http://etats.developpez.com
Lien18 : http://wiki.apache.org/xmlgraphics-fop/HowTo/PHPJavaBridge
Lien19 : http://dynamicjasper.sourceforge.net/
Lien20 : http://php.developpez.com/etudes/livre-blanc-php-entreprise/
Lien21 : http://google.fr/trends?q=java,php,python,.net&ctab=0&geo=all&date=all&sort=0
Lien22 : http://www.agata.org.br/
Lien23 : http://groups.google.fr/group/comp.lang.php/browse_thread/thread/b9e178b19c135cca/ba59aa035b6e4b8f
Lien24 : http://www.zend.com/fr/products/zend_platform/what_s_new
Lien25 : http://www.eclipse.org/birt/phoenix/
Lien26 : http://birtworld.blogspot.com
Lien27 : http://php-java-bridge.sourceforge.net/pjb/index.php
Lien28 : http://www.clever-age.com/veille/blog/retour-d-experiences-sur-php-java-bridge.html
Lien29 : http://charly-clairmont.developpez.com/tutoriels/php-jasper-reports/
Lien30 : http://prototypejs.org/
Lien31 : http://siddh.developpez.com/articles/ajax/#LIII
Lien32 : http://prototypejs.org/api/event
Lien33 : http://dcabasson.developpez.com/articles/javascript/ajax/documentation-prototype-1.4.0/#L5-A-7
Lien34 : http://amillet.developpez.com/tutoriels/javascript/ajax-prototype/request.html
Lien35 : http://amillet.developpez.com/tutoriels/javascript/ajax-prototype/updater.html
Lien36 : http://amillet.developpez.com/tutoriels/javascript/ajax-prototype/periodicalUpdater.php
Lien37 : http://javascript.developpez.com/faq/?page=Ajax#ajax.json.introduction
Lien38 : http://mike.teczno.com/json.html
Lien39 : http://ajax.developpez.com/
Lien40 : http://dcabasson.developpez.com/articles/javascript/ajax/documentation-prototype-1.4.0/
Lien41 : http://siddh.developpez.com/articles/ajax/
Lien42 : http://amillet.developpez.com/tutoriels/javascript/ajax-prototype/#LI
Lien43 : http://mehdi-fekih.developpez.com/articles/dotnet/dictionnaires
Lien44 : http://dotnet.developpez.com/livres/?page=livresASPNET#L0764584642
Lien45 : http://weblogs.asp.net/scottgu/archive/2007/10/03/releasing-the-source-code-for-the-net-framework-libraries.aspx
Lien46 : http://blog.developpez.com/index.php?blog=161&title=le_framework_net_3_5_open_source&more=1&c=1&tb=1&pb=1
Lien47 : http://man.developpez.com/man1/ddd.1.php
Lien48 : http://www.gnu.org/manual/ddd/html_mono/ddd.html
Lien49 : http://hiko-seijuro.developpez.com/articles/ddd/
Lien50 : http://library.gnome.org/devel/gtk/unstable/GtkTooltip.html
Lien51 : http://library.gnome.org/devel/gtk/unstable/GtkBuilder.html
Lien52 : http://gtk.developpez.com/outils/#glade
Lien53 : http://library.gnome.org/devel/gtk/unstable/GtkScaleButton.html
Lien54 : http://library.gnome.org/devel/gtk/unstable/GtkVolumeButton.html
Lien55 : http://library.gnome.org/devel/gtk/unstable/GtkRecentAction.html
Lien56 : http://mail.gnome.org/archives/gtk-devel-list/2007-September/msg00052.html
Lien57 : http://library.gnome.org/devel/glib/unstable/glib-Sequences.html
Lien58 : http://library.gnome.org/devel/glib/unstable/glib-Perl-compatible-regular-expressions.html
Lien59 : http://mail.gnome.org/archives/gtk-devel-list/2007-August/msg00011.html
Lien60 : http://blog.developpez.com/index.php?blog=58&title=gtk_2_12_0&more=1&c=1&tb=1&pb=1
Lien61 : http://silkyroad.developpez.com/vba/tableaux/
Lien62 : http://argyronet.developpez.com/office/access/runtime/
Lien63 : http://argyronet.developpez.com/office/access/runtime/2003
Lien64 : http://argyronet.developpez.com/office/access/runtime/2007/
Lien65 : http://access.developpez.com/faq/

Lien66 : http://www.w3.org/
Lien67 : http://ietab.mozdev.org/
Lien68 : http://lgmorand.developpez.com/articles/extensions-firefox/#L2.2.6
Lien69 : http://www.microsoft.com/downloads/details.aspx?FamilyID=E59C3964-672D-4511-BB3E-2D5E1DB91038&displaylang=en
Lien70 : http://www.w3.org/WAI/
Lien71 : http://www.access-board.gov/508.htm
Lien72 : http://lgmorand.developpez.com/articles/ie-developper-toolbar/
Lien73 : http://blog.developpez.com/index.php?blog=134&title=xp_durera_plus_longtemps_que_prevu&more=1&c=1&tb=1&pb=1
Lien74 : http://developer.apple.com/documentation/GraphicsImaging/Conceptual/QuartzComposer/index.html
Lien75 : http://developer.apple.com/tools/interfacebuilder.html
Lien76 : http://www.apple.com/macosx/features/automator/
Lien77 : http://www.sam.eliotis.com/
Lien78 : http://sebastien-marchand.developpez.com/tutoriels/mac/smart-automation-maker/videos/DemoBlueBoxCreation.mov
Lien79 : http://developer.apple.com/documentation/Cocoa/Conceptual/ObjectiveC/ObjC.pdf
Lien80 : http://developer.apple.com/cocoa/
Lien81 : http://developer.apple.com/documentation/Cocoa/
Lien82 : http://sebastien-marchand.developpez.com/tutoriels/mac/smart-automation-maker/
Lien83 : http://mac.developpez.com/faqs/mac/
Lien84 : http://trac.edgewall.org/
Lien85 : http://subversion.tigris.org/
Lien86 : ftp://ftp-developpez.com/miles/tutoriels/hardware/NAS/TRAC-Subversion/fichiers/syno-telnet-r4.zip
Lien87 : ftp://ftp-developpez.com/miles/tutoriels/hardware/NAS/TRAC-Subversion/fichiers/ds101-bootstrap_1.0-4_armeb.xsh
Lien88 : ftp://ftp-developpez.com/miles/tutoriels/hardware/NAS/TRAC-Subversion/fichiers/ds101-bootstrap_1.0-4_powerpc.xsh
Lien89 : http://hugo.developpez.com/tutoriels/outils/subversion/
Lien90 : ftp://ftp-developpez.com/miles/tutoriels/hardware/NAS/TRAC-Subversion/fichiers/S80tracd
Lien91 : ftp://ftp-developpez.com/miles/tutoriels/hardware/NAS/TRAC-Subversion/fichiers/trac-post-commit-hook.py
Lien92 : http://miles.developpez.com/tutoriels/hardware/NAS/TRAC-Subversion/
Lien93 : http://blog.developpez.com/index.php?blog=142&title=les_imacs_sont_ils_true_colors_ou_pas&more=1&c=1&tb=1&pb=1
Lien94 : http://forums.anandtech.com/messageview.aspx?catid=31&threadid=2049206
Lien95 : http://baptiste-wicht.developpez.com/tutoriel/hardware/choix/ecran/
Lien96 : http://phphaml.sourceforge.net/
Lien97 : http://haml.hamptoncatlin.com/download/
Lien98 : http://groups.google.com/group/haml/web/syntax-highlighting
Lien99 : http://www.developpez.net/forums/forumdisplay.php?f=235
Lien100 : http://haml.hamptoncatlin.com/
Lien101 : http://groups.google.com/group/haml/
Lien102 : http://thomas-brian.developpez.com/articles/haml/
Lien103 : http://ruby.developpez.com/livres/#L274402127X
Lien104 : http://ruby.developpez.com/livres/#L9782212120790
Lien105 : http://ruby.developpez.com/livres/#L2841773884
Lien106 : http://martinfowler.com/articles/mocksArentStubs.html
Lien107 : http://forum.wordreference.com/showthread.php?t=31170
Lien108 : http://dannorth.net/introducing-bdd/
Lien109 : http://martinfowler.com/bliki/ObjectMother.html
Lien110 : http://www.amazon.com/exec/obidos/ASIN/020161622X
Lien111 : http://martinfowler.com/bliki/RoleInterface.html
Lien112 : http://bruno-orsier.developpez.com/mocks-arent-stubs/
Lien113 : http://msdn2.microsoft.com/en-us/library/microsoft.sharepoint.speventreceiverbase.disableeventfiring.aspx
Lien114 : http://msdn2.microsoft.com/en-us/library/microsoft.sharepoint.speventreceiverbase.enableeventfiring.aspx
Lien115 : http://www.u2u.info/Blogs/Patrick/Lists/Posts/Post.aspx?ID=1547
Lien116 : http://stephaneey.developpez.com/tutoriel/sharepoint/eventhandler/download/DemoEventHandler.zip
Lien117 : http://stephaneey.developpez.com/tutoriel/sharepoint/eventhandler/
Lien118 : http://sharepoint.developpez.com/faq