Developpez

Le Mag

Edition d'Avril-Mai 2009.
Numéro 21.
Magazine en ligne gratuit.

Réalisation : Alexandre Pottiez
Rédaction : la rédaction de Developpez
Contact : magazine@redaction-developpez.com

Sommaire

Editorial

Dans le souci de toujours vous satisfaire,
découvrez dès ce mois-ci le nouveau look de votre magazine préféré!

Retrouvez une sélection de nos meilleurs articles, critiques de livres, et questions/réponses sur diverses technologies.

Profitez-en bien !

La rédaction

Article C/C++/GTK

Boost 1.38 et MingW : un mariage qui prend du temps !

Résoudre les dysfonctionnements lors de l'utilisation des dates et des timers.

par **3DArchi**
Page 30

Article Qt & 2D/3D/Jeux

Integrer Ogre à Qt - Partie 2

Retrouvez la seconde partie de ce tutoriel destiné à illustrer l'intégration du moteur 3D Ogre à Qt.

par **Denys Bulant**
Page 63

Java / Eclipse

La FAQ JavaFX

Qu'est ce que JavaFX ?

Tout d'abord, il faut savoir qu'officiellement, le nom exact de ce langage n'est pas JavaFX mais JavaFX Script. Mais nous parlerons toujours ici dans cette FAQ de JavaFX car nous trouvons, personnellement, que de parler de JavaFX Script risquerait de mêner à la confusion avec le langage JavaScript que nous retrouvons dans tous nos navigateurs Web.

JavaFX est un langage de script, fonctionnel, qui nécessite une compilation préalable avant de pouvoir être exécuté.

Le langage JavaFX s'appuye sur la Machine Virtuelle de Java (JVM) pour s'exécuter. Et peut également tirer profit de la richesse des API écrites en Java et disponibles actuellement.

Le code de votre source doit se trouver dans un fichier ayant comme suffixe .fx et le résultat de la compilation de ce code source sera un ou plusieurs fichier .class,

Bien qu'il soit possible d'écrire des applications en JavaFX n'ayant aucun GUI, JavaFX a été pensé tout particulièrement pour faire des applications ayant un GUI très graphique et fortement animé.

C'est la raison pour laquelle on retrouve dans les API de base de JavaFX tout ce qu'il faut pour faire des animations et des manipulations d'objets graphiques en tout genre.

Dois-je connaître le langage Java ou l'API Java avant de me lancer dans JavaFX ?

Non, mais cela est fortement recommandé car vous ne pourrez pas aller bien loin uniquement avec l'API JavaFX 1.0.
Si vous vous contentez de faire des applications graphiques (similaires à du Flash par exemple), vous pouvez vous contenter de JavaFX uniquement.

Puis-je appeler directement du Java depuis JavaFX ?

Oui, c'est même recommandé car l'API JavaFX 1.0 est somme toute assez limitée. Vous devrez donc vous reposer sur les bibliothèques Java existantes pour étendre les possibilités de votre application. Vous avez directement accès à tout ce qui est présent sur le CLASSPATH.

Tout comme en Java, vous pouvez soit directement indiquer le nom complet (package+nom) de la classe :

```
println(java.lang.System.getProperty("java.versio
n"));
```

Soit importer l'intégralité du package :

```
import java.lang.* ;

println(System.getProperty("java.version"));
```

Soit importer directement la classe elle-même :

```
import java.lang.System ;

println(System.getProperty("java.version"));
```

Comme vous l'aurez remarqué, il est nécessaire d'importer manuellement les classes du package java.lang

Que sont les fichiers FXD, FXZ et FXM ?

Un fichier .FXZ (**JavaFX Zipped Asset File**) est une archive au format ZIP qui contient un fichier FXD (**JavaFX Description File**) qui lui contient du JavaFX Script.
Cette archive peut également inclure des images bitmaps ou des fichiers de polices de charactères qui sont utilisées dans le design.

Il existe également des fichiers .FXM (**JavaFX MultiMedia File**) contenant des séquences vidéo encodées avec le codec On2 et du son encodé en MP3.

Retrouvez ces questions et de nombreuses autres sur la FAQ JavaFX : Lien1

Les dernières news

La rubrique Java héberge désormais des sites web Java

Bonjour,

Après la mise en ligne et l'enrichissement de JavaSearch (Lien2) (idée originale de Laurent Perron, maintenu par Baptiste Wicht), l'équipe Java a le plaisir de vous annoncer ce nouveau service que nous mettons à dispositions gratuitement à nos partenaires et contributeurs bénéficiant déjà d'un hébergement classique (Lien3).

Parmi les heureux bénéficiaires, on retrouve notamment 2 JUGs :

- Le Tours JUG (Lien4)
- Le Lyon JUG (Lien5)

Ces 2 JUGs ont choisi XWiki comme outil, et l'équipe Java leur met à disposition, tout comme le fait XWiki SAS pour l'ensemble des JUGs, à la seule nuance près que nous ne nous limitons (limiterons) pas à cette technologie et accompagnerons ces communautés tout au long de leur évolution et ceci quelle que soit la solution logicielle retenue (pourvu qu'elle repose sur Java). Avis aux autres JUGs francophones naissants ou déjà existants, n'hésitez pas à nous contacter via le forum ou par email.

Mais l'hébergement Java prend également une autre forme : celle de la mise en ligne d'applications de type **démo**
Vous trouverez une première démo illustrant l'article CRUD avec JSF et JPA de Jawher Moussa (Lien6) à l'adresse suivante : (Lien7) (ne cherchez pas, il n'y a rien pour l'instant sur la home du domaine demo). Cette possibilité sera désormais offerte à tous les rédacteurs de l'équipe Java souhaitant illustrer leurs publications par des exemples concrets.

Je tiens tout particulièrement à remercier lunatix (Lien8), administrateur bénévole de cet hébergement, sans qui tout ceci ne serait possible, ainsi que les membres de l'équipe bénévole Java (Lien9) participant à ce projet, mais également les administrateurs de Developpez.com pour l'aide à la mise en place.

Si vous avez des remarques constructives, l'envie de participer à ce projet en apportant vos idées et votre aide, n'hésitez pas à revenir vers nous que ce soit ici, par message privé sur le forum, ou par mail, car nous ne nous arrêterons bien sur pas là

Commentez cette news en ligne : Lien10

Les livres Java

Java EE - Guide de développement d'applications web en Java

Critique du livre par Celinio Fernandes

Ce livre s'adresse à des débutants mais aussi à des personnes experimentées car il contient de nombreux rappels et aussi des astuces.

Le chapitre 1 couvre des choses qui doivent être connues lorsque l'on veut faire du développment Java EE. Cela va de l'installation du JDK à la présentation du modèle MVC, en passant par l'installation et la configuration d'un serveur Tomcat dans Eclipse et sous Windows.

Le chapitre 2 présente l'installation de Tomcat 6.0.X sous Linux, ainsi que l'installation et la configuration du serveur Web Apache en frontal du serveur Tomcat. Les fichiers XML de configuration des 2 serveurs (server.xml, web.ml, tomcat-users.xml, etc) sont expliqués, avec en prime des rappels sur XML. La sécurité, la gestion des utilisateurs, le connecteur HTTPS sont également détaillés. Ce chapitre se veut pratique puisqu'il détaille la création et le déploiement d'un projet dans Tomcat. Le monitoring et les tests de montée en charge sont abondamment expliqués. J'ai particulièrement apprécié le cas de tests s'appuyant sur JMeter.

Le chapitre 3 présente les JSP. Elles sont utilisées dans l'élaboration de l'application BetaBoutique. C'est en toute logique que les tags JSTL sont ensuite présentés, et sont suivis d'un sous-chapitre consacré à la création de taglibs personnalisées. En fait, ce chapitre fait le tour complet du développement avec des JSP (avec des servlets, des JavaBeans, etc ...). J'ai apprécié l'utilisation de la librairie commons-beanutils pour la manipulation des JavaBeans ainsi que du parser XStream pour la sérialisation/déserialisation d'objets Java.

Le chapitre 4 est consacré aux servlets. Une maquette de l'application BetaBoutique est insérée dans ce chapitre. L'application commence à prendre forme avec l'utilisation des servlets (sessions, filtres, cookies, interface RequestDispacter, etc ...).

Le chapitre 5 présente l'API JDBC. La base de données de l'application est construite à partir de MySQL. Le partage d'une connexion est bien illustré par du code. L'interface DataSource est ensuite utilisée et ses avantages expliqués. Un pool de connexion est créé dans Tomcat. De nouvelles astuces sont introduites (JavaScript, Ajax, la librairie DbUtils). Ce chapitre sur les bases de données serait incomplet s'il ne traitait pas des transactions, ce qui est chose faite juste après. Finalement l'authentification est étudiée, via la base de données et via les fichiers de configuration.

Le chapitre 6 décrit le framework Struts dans sa version 1.X, à travers le développement de l'application chatbetaboutique. Les caractéristiques les plus connues de ce framework sont mises en exergue : action, form, dynaform, internationalisation, validations avec validators et javascript. J'ai apprécié l'illustration par du code des actions les moins connues mais pourtant bien utiles : DispatchAction, LookUpDispatchAction, MappingDispatchAction, SwitchAction. Le module administration de cette application est par la suite bien décrit et implémenté.

Enfin le dernier chapitre détaille la très utilisée librairie log4J, la construction de tâches avec ANT, l'optimisation de la mémoire dans Tomcat 6.X.

En résumé, j'ai trouvé le livre très riche, surtout pour les débutants, et j'avoue que personnellement j'aurais souhaité avoir un tel livre dans mes étagères il y a quelques années. Il représente également une source importante de rappels pour les plus experimentés.

Le code source du livre est bien entendu disponible sur le site des éditions ENI, ainsi que sur http://www.gdawj.com/ (Lien11)

Retrouvez ces critiques de livre sur la page livres Java : Lien12

Création d'un composant facelet personnalisé avec Facelets JSF et Richfaces

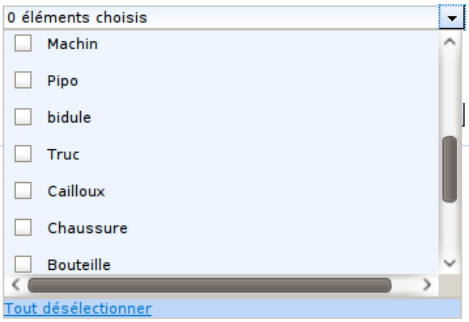
Si vous utilisez JSF, vous avez dû remarquer qu'on fait souvent des assemblages similaires de composants.

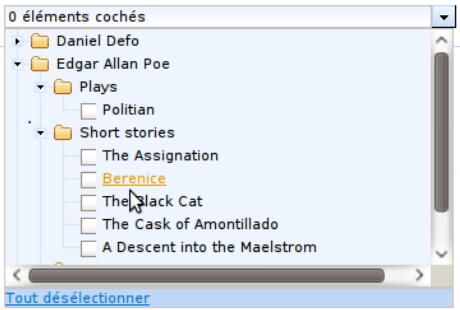
Parfois l'idée vous passe par la tête de créer votre propre composant JSF pour remplacer ces assemblages redondants, mais créer un composant JSF est long et difficile.
La difficulté réside notamment dans la gestion du cycle de vie JSF du composant que vous créez.
Bien heureusement il existe une solution alternative grâce à Facelets
En effet vous pouvez créer un assemblage de composants JSF qui deviendra alors un composant en soit, mais un composant Facelet cette fois.
Ce qui ne changera en rien son utilisation.

1. Introduction

Le but de ce tutoriel est de de créer une list box qui peut recevoir n'importe quoi dans la liste déroulante, dans mon cas j'ai besoin de listes ou d'arbres de checkbox avec label.
J'ai donc besoin d'utiliser dans ma liste déroulante, soit une rich:dataTable soit un rich:tree.

Je vous renvoie au Tutoriel de L. Mellouk sur richfaces (Lien13) l'utilisation du rich:tree ou de la rich:dataTable.
On souhaite arriver à un composant générique qui permettrait de faire ce genre de choses :





2. Les étapes

Créer un composant Facelets requiert trois étapes :

- Création d'un fichier de taglib pour pouvoir le déclarer à facelets et l'utiliser.
- Déclaration du fichier de taglib à facelets dans le web.xml.
- Création du composant dans un fichier xhtml.

3. Configuration

Le plus simple pour intégrer les composants que vous créez à votre WAR est de les placer dans le repertoire WEB-INF
Par exemple j'ai crée un répertoire components dans WEB-INF.
Voyons maintenant la configuration du composant, tout d'abord dans web.xml :
On indique à Facelets d'allerchercher les composants additionnels dans WEB-INF/facescomponents.taglib.xml :

Ajout dans web.xml

```
<context-param>
        <param-name>facelets.LIBRARIES</param-
name>
        <param-value>/WEB-
INF/facescomponents.taglib.xml</param-value>
</context-param>
```

4. Taglib

Ensuite, comme vous pouvez le voir précédemment on créer un fichier taglib pour notre composants :
voici le fichier facescomponents.taglib.xml :

facescomponents.taglib.xml

```
<?xml version="1.0"?>
<!DOCTYPE facelet-taglib PUBLIC
  "-//Sun Microsystems, Inc.//DTD Facelet Taglib
1.0//EN"
  "facelet-taglib_1_0.dtd">
<facelet-taglib>
    <namespace>http://www.dreamisle.net/facescomp
onents</namespace>
  <tag>
    <tag-name>selectListCheckBox</tag-name>
    <source>components/selectListCheckBox.xhtml</
source>
  </tag>
</facelet-taglib>
```

5. Création du composant

On peut maintenant créer le composant.

J'ai fourni CSS/Javascript volontairement au cas ou vous souhaitiez réutiliser ce composant.
Néanmoins, tout ce qui correspond à la position et au z-

index est important pour permettre la création de la pseudo liste déroulante.
Le javascript (ici fait avec jQuery) est utilisé pour enrouler/dérouler la liste, et pour compter les checkbox cochées dans le contenu.
Il s'agit d'un exemple de comportement javascript que l'on peut attribuer à un composant, à vous d'adapter à votre choix. J'ai choisi d'utiliser jQuery pour plus de portabilité.
De plus la librairie permet d'éviter de polluer le code jsf d'appels javascript sur les “onclick”, “onchange” etc …

Vous pouvez bien sur définir ce que vous voulez dans le comportement de votre composant, Voire même contrôler le composant avec un objet Seam appellé en ajax, mais cela le rend inutilisable en dehors de votre contexte Seam.
Le système d'id est un peu compliqué, en fait on créer nos ids à partir de l'id du composant créé dans la JSF, pour éviter les erreurs de duplicate ID.
Donc les ids sont faits avec des EL, mais ne vous formalisez pas de cela.
Cela complexifie la lecture du code jQuery et JSF mais c'est une façon simple d'éviter le duplicate ID si vous insérez plusieurs fois le composant dans la même page.

le composant en lui même : selectListCheckBox.xhtml

```
<!DOCTYPE composition PUBLIC "-//W3C//DTD XHTML
1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">

<ui:composition
xmlns="http://www.w3.org/1999/xhtml"
    xmlns:s="http://jboss.com/products/seam/tagli
b"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:rich="http://richfaces.org/rich"
    xmlns:a4j="http://richfaces.org/a4j">

    <!-- Style-->
          <style>

                        /* on met le composant en
position relative, pour pouvoir placer la liste
déroulante au dessus en absolute après*/
            .divLayoutSelectListCheckBox {
              position: relative;
            }
                        /* la largeur totale du
composant*/
            .dataGridLayoutSelectListCheckBox {
              width:350px
            }
            /* le champ input simulant une select
box */
            .idropDownSelectListCheckBoxInput {
                width:332px;
                margin:0px;
                z-index: 2;
            }
                        /* uniquement pour que la
rich:dataTable n'aie pas de bordures. */
            .dataTableDropDownList {
              width: 350px;
              border-style: none;
              background-color: #EEF5FE;
            }
            /* uniquement pour que les colonnes
de la rich:dataTable n'aient pas de bordures. */
            .dataTableDropDownListColumn {
              border-bottom: 0px;
              border-right: 0px;
            }
            /* le div de la liste en elle meme.
*/
            .dropDownSelectListCheckBox {
              height: 202px;
              overflow: auto;
              background-color: #EEF5FE;
              border: 1px solid #CAC5BE;
            }
            /* on fixe en position absolue et
avec un z-index important  le div entourant la
liste déroulante pour qu'elle passe au dessus du
reste*/
            .fullListDropDownCheckBox {
              position: absolute;
              z-index: 5;
            }
                        /* le pied de la liste
deroulante*/
            .dropDownSelectListCheckBoxFooter {
              background-color: #BED6F8;
              border: 1px solid #CAC5BE;
            }

            </style>
                        <!-- Javascript for the
component behaviour-->
            <a4j:loadScript
src="resource://jquery.js"/>
            <script type="text/javascript">
                    jQuery(document).ready(functi
on() {
                        /* par défaut on cache la
liste déroulante */
                        jQuery("##{id}magicBoxLis
t").hide();



                        /* si le paramètre footer
est à faux on cache le footer*/
                        if (!jQuery(#{footer})) {
                            jQuery("##{id}footerL
istBox").hide();
                        }
                                                /
* on remplit la valeur du champ input par défaut
*/
                        jQuery("##{formId}\\:#{id
}statesinput").attr("value", "0" + " " +
'#{selectedLabel}');

                        /* un click dans l'input
cache ou montre la liste*/
                        jQuery("##{formId}\\:#{id
}statesinput").click(function() {
                            if
(jQuery("##{id}magicBoxList").is(":hidden")) {
                                jQuery("##{id}mag
icBoxList").show();
                            } else {
                                jQuery("##{id}m
agicBoxList").hide();
                            }
                        });
                        /* un click sur la flèche
cache ou montre la liste.*/
                        jQuery("##{formId}\\:#{id
}showandhidelinkid").click(function() {
```

```
                            if
(jQuery("##{id}magicBoxList").is(":hidden")) {
                                jQuery("##{id}mag
icBoxList").show();
                            } else {
                                jQuery("##{id}m
agicBoxList").hide();
                            }
                        });



                        /* désélectionne toutes
les checkbox lorsqu'on clique sur le lien
correspondant*/
                        jQuery('##{formId}\\:#{id
}deselectAllButtonFooter').click(function(){
                            jQuery("##{id}dropLis
tContainer
input[@type='checkbox']").attr('checked', false);
                            jQuery("##{formId}\\:
#{id}statesinput").attr("value", "0 " +
'#{selectedLabel}');
                        });

                        /*  met  à jour le
compteur de cases cochées à chaque click sur une
checkbox  */
                        jQuery("##{id}dropListCo
ntainer
input[@type='checkbox']").click(function() {
                            count =
jQuery("##{id}dropListContainer
input[@type='checkbox']:checked").length;
                            if (count >
#{maxCheck}) {
                                jQuery("##{formId
}\\:#{id}statesinput").attr("value",
"#{maxSelectMessage}" + "  " + "#{maxCheck}" + "
éléments");
                                jQuery(this).attr
('checked', false);
                            } else {
                                jQuery("##{formI
d}\\:#{id}statesinput").attr("value", count + " "
+ '#{selectedLabel}');
                            }
                        });
                    });
            </script>

<!-- le code proprement dit du composant -->
        <h:panelGrid columns="2">
            <s:div
styleClass="divLayoutSelectListCheckBox"
id="#{id}idPanelGridForFc">
                <h:panelGrid columns="2"
border="0" cellpadding="0" cellspacing="0"
                    styleClass="dataGridLayoutSelec
tListCheckBox" >
                    <!-- on imite une select
box avec un inputText desactivé et une image de
flèche -->
                    <h:inputText
id="#{id}statesinput" readonly="true"
                        styleClass="idropDownSelec
tListCheckBoxInput" />
                    <a4j:commandLink
id="#{id}showandhidelinkid">
                        <h:graphicImage
value="/img/arrow.png" alt="&gt;"/>
                    </a4j:commandLink>
                </h:panelGrid>
                <h:panelGroup>
                <!-- la liste déroulante-->
```

```
                    <div
class="fullListDropDownCheckBox"
id="#{id}magicBoxList">
                    <div
class="dropDownSelectListCheckBox"
id="#{id}dropListContainer">
                    <!-- le contenu de la liste
déroulante sera inséré ici-->
                        <ui:insert />
                    </div>
                    <!-- on ajoute un footer à la
liste déroulante avec un lien  "tout
déselectionner"-->
                    <div
class="dropDownSelectListCheckBoxFooter"
id="#{id}footerListBox">
                        <a4j:commandLink
id="#{id}deselectAllButtonFooter">
                            Tout désélectionner
                        </a4j:commandLink>
                    </div>
                    </div>
                </h:panelGroup>
            </s:div>
        </h:panelGrid>

</ui:composition>
```

Comme vous pouvez le voir, on a créé le composant de la même manière que l'on écrirait une page jsf classique, la différence réside uniquement dans le fait que l'on crée une composition.

L'élément important ici est le **<ui:insert/>**
C'est ici que sera placé le contenu de la liste déroulante, remplie par l'utilisateur du composant dans une page JSF.
Autre élément à noter: les EL utilisées dans le composant.
Par exemple ici **#{id}** : il s'agit en fait de récupérer les paramètres, passés au composant de manière classique dans la JSF.
Pour mieux comprendre lisez la partie "utilisation du composant" vous allez tout de suite voir que les paramètres déclarés au composant, sont récupérables sous forme d'EL avec la même casse que dans l'EL, ainsi votre composant est facilement paramétrable.

#{formId}, **#{maxSelectMessage}**, **#{maxCheck}**, **#{footer}** et **#{selectedLabel}** ne sont utilisés que dans les fonctions de comportement jQuery du composant.
Vous voyez que vous pouvez même utilisez les paramètres pour définir le comportement du composant avec javascript.

J'ai choisi pour l'exemple d'utiliser des **a4j:commandLink** liés à des appels javascripts (déclarés dans la partie javascript) mais il y a bien d'autres façons de faire.

6. Utilisation du composant

Et voici maintenant un exemple d'utilisation qui permet d'arriver aux captures du début de cet article :

```
<!DOCTYPE composition PUBLIC "-//W3C//DTD XHTML
1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<ui:composition
xmlns="http://www.w3.org/1999/xhtml"
    xmlns:s="http://jboss.com/products/seam/tagli
```

```
b"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:rich="http://richfaces.org/rich"
    xmlns:a4j="http://richfaces.org/a4j"
    xmlns:fc="http://www.meteojob.com/facescompon
ents"
    template="layout/template.xhtml">

  <ui:define name="body">
    <rich:panel>
    <f:facet name="header"> select test
</f:facet>
    <h:form id="formRichListId">
    <!-- ici une liste avec en label une chaine
de caracteres, et devant le label une checkbox --
>
      <fc:selectListCheckBox id="fcSelectList"
formId="formRichListId" selectedLabel="éléments
choisis"
         maxCheck="4" maxSelectMessage="Vous
devez choisir au maximum:" footer="true">
              <rich:dataTable
value="#{dreamSelectBox.theList}" var="item"
                styleClass="dataTableDropDownLis
t" >
                <rich:column
styleClass="dataTableDropDownListColumn">
                  <h:selectBooleanCheckbox
value="false" name="checkboxRichList" />
                </rich:column>
                <rich:column
styleClass="dataTableDropDownListColumn">
                  <h:outputText value="#{item}"
escape="false" />
                 </rich:column>
              </rich:dataTable>
      </fc:selectListCheckBox>

      <rich:spacer height="50" />
        <!-- ici un arbre avec aussi des
checkboxs, j'ai
        repris l'arbre donné en exemple dans la
démo en ligne de richfaces -->
      <fc:selectListCheckBox id="fcSelectTree"
formId="formRichListId"
         selectedLabel="éléments cochés"
         maxCheck="4" maxSelectMessage="Vous
devez choisir au maximum:" footer="true">
              <rich:tree style="width:350px;"
nodeSelectListener="#{dreamSelectBox.theTree.proc
essSelection}"
                reRender="selectedNode"
ajaxSubmitSelection="true"  switchType="client"
                value="#{dreamSelectBox.theTree.g
etTreeNode()}" var="item" ajaxKeys="#{null}" >
              <rich:treeNode>
                <f:facet
name="iconLeaf"><h:selectBooleanCheckbox
value="false"/></f:facet>
                <h:outputText value="#{item}" />
              </rich:treeNode>
              </rich:tree>
      </fc:selectListCheckBox>

      </h:form>
    </rich:panel>
</ui:define>

</ui:composition>
```

Et voilà, vous avez votre composant, j'espère que cet article aura été utile.

Retrouvez l'article de Mikael Robert en ligne : Lien14

Introduction à PHPExcel

Nous allons voir dans cet article une bibliothèque orientée objet (PHP 5), la génération de feuilles de calcul sous différents formats d'exportation et/ou d'enregistrement.
Cette bibliothèque étant assez conséquente, et mon apprentissage de celle-ci ne faisant que de commencer, nous aurons un article construit au fil du temps.

1. Avant-propos

Lors de la rédaction de l'article sur Spreadsheet_Excel_Writer en PHP (Lien15), Yogui , m'avait demandé si j'allais faire un parallèle avec la classe PHPExcel (Lien16). Ne connaissant pas cette classe, je n'avais pas répondu positivement à cette demande.
Et je ne vais toujours pas le faire ici, parce que nous ne sommes pas à faire un parallèle, mais bien à découvrir un outil puissant qui va nous rendre de grands services dans la génération de nos tableurs.

1.1. PHPExcel

PHPExcel va nous proposer la génération des formats suivants :

- Excel 2007
- Excel 2007 - compatible Excel 2003
- Excel 5 - via la librairie Spreadsheet_Excel_Writer (Lien17)
- PDF - via la librairie FPDF (Lien18)
- CSV
- HTML

En plus, PHPExcel va affiner nos tableurs avec des éléments tel que :

- Formater les cellules.
- Ajouter des formules
- Ajouter des images
- Protéger les données, les cellules
- ....

1.1.1. Téléchargement

Pour commencer, nous allons télécharger la librairie PHPExcel (Lien19), ce fichier comporte la librairie dans le répertoire ./Classes, nous avons également l'API dans le répertoire ./Documentation , ainsi qu'un répertoire ./Tests avec un certain nombre d'exemples.
Il faut extraire le répertoire ./Classes et le déposer dans le répertoire de votre Web serveur.

1.1.2. Spécificités

PHPExcel ne fonctionne qu'avec PHP 5 et réclame des extensions spécifiques pour fonctionner correctement. Ces extensions sont :

- Zip
- GD
- XML

Vous pouvez vérifier que vous avez ces extensions avec la fonction.

```
phpinfo();
```

1.1.3. Testé sur ....

J'ai testé tous les exemples sur différents OS et tableurs.

- O0o 2.4 / Debian
- O0o 3.0 / Debian
- Excel 2007 /VirtualBox/Windows XP
- Excel 2007 /Windows Serveur 2003

On peut retrouver quelques différences entre les versions de tableurs, notamment au niveau des couleurs de remplissage des cellules, ou encore dans les bordures, Ooo 2.4 ne reconnaît pas les formats type Dash Dot (BORDER_DASHDOT).
Sur le serveur 2003 en production avec PHP 5.2.6 , j'ai été confronté à un souci de dll ("php_zip.dll"), cela tronque le fichier.

Sur le forum de PHPExcel il est conseillé de télécharger une autre version de php_zip.dll, notamment sur http://snaps.php.net (Lien20)

2. Premier fichier

Nous allons voir comment créer notre premier fichier au format Excel 2007

2.1. MaitrePylos en A1

```
include 'PHPExcel.php';
include 'PHPExcel/Writer/Excel2007.php';

$workbook = new PHPExcel;

$sheet = $workbook->getActiveSheet();
$sheet->setCellValue('A1','MaitrePylos');

$writer = new
PHPExcel_Writer_Excel2007($workbook);

$records = './fichier.xlsx';

$writer->save($records);
```

Ce code enregistre un fichier nommé 'fichier.xlsx' sur le disque dur du serveur, il n'apparaîtra pas sur le navigateur.

Il faut d'abord inclure les fichiers nécessaires à la génération du tableur

```
include 'PHPExcel.php';
include 'PHPExcel/Writer/Excel2007.php';
```

Ensuite nous instancions notre objet PHPExcel

```
$workbook = new PHPExcel;
```

Nous activons la feuille sur laquelle nous allons travailler (la feuille par défaut), grâce à la méthode ->getActiveSheet().

```
$sheet = $workbook->getActiveSheet();
```

Lors de l'instanciation de notre objet PHPExcel, le constructeur a également instancié diverses classes, notamment la classe PHPExcel_Worksheet
et la méthode getActiveSheet nous donnes donc la main sur cet objet.

Nous remplissons notre première cellule avec la méthode ->setCellValue()

```
$sheet->setCellValue('A1','MaitrePylos');
```

Vous remarquerez que, pour désigner la cellule, la méthode ->setCellValue() accepte son nom de tableau à double entrée.
Il existe une autre méthode pour désigner la cellule que l'on veut modifier.

```
$sheet->setCellValueByColumnAndRow(1, 4,
'MaitrePylos');
```

Cette méthode prend trois paramètres :

- La ligne
- La colonne
- La valeur

Notez que dans le cas de cette méthode, la ligne commence à partir de zéro(0), et la colonne commence à un (1).

Enfin pour créer le fichier, nous devons instancier un objet writer, spécifique au type de tableau que nous voulons générer.

```
$writer = new
PHPExcel_Writer_Excel2007($workbook);
```

Nous donnons un nom à notre fichier, et l'enregistrons

```
$records = './fichier.xlsx';
$writer->save($records);
```

Vous prêterez une attention toute particulière à l'extension du fichier, en l'occurrence ici "xlsx".

2.2. Dans un autre format

Pour enregistrer le fichier dans un autre format, il faut juste 3 modifications au script précédent

2.2.1. Excel5

```
include 'PHPExcel/Writer/Excel5.php';
...
$writer = new PHPExcel_Writer_Excel5();
...
$records = './fichier.xls';
```

2.2.2. PDF

```
include 'PHPExcel/Writer/PDF.php';
...
$writer = new PHPExcel_Writer_PDF();
$writer->setSheetIndex(0);//Une seule feuille
possible
...
$records = './fichier.pdf';
```

Dans la génération des fichiers de types PDF, HTML, il n'est possible que de travailler sur une seule feuille, et nous devons lui indiquer laquelle, via la méthode ->setSheetIndex()

2.2.3. HTML

```
include 'PHPExcel_Writer/HTML.php';
...
$writer = new PHPExcel_Writer_HTML();
$writer->setSheetIndex(0);//Une seule feuille
possible

...
$records = './fichier.html';
```

2.2.4. CSV

```
include 'PHPExcel/Writer/CSV.php';
...
$writer = new PHPExcel_Writer_CSV();
$writer->setDelimiter(",");//l'opérateur de
séparation est la virgule
$writer->setSheetIndex(0);//Une seule feuille
possible

...
$records = './fichier.csv';
```

Dans le cadre d'un fichier de type CSV, il faut en plus noter le séparateur, ce qui est le rôle de la méthode ->setDelimiter().

2.2.5. Excel 2007 - compatible 2003

Pour assurer une compatibilité avec Excel 2003, il suffit de mettre la méthode ->setOffice2003Compatibility(), à true.

```
$writer = new
PHPExcel_Writer_Excel2007($workbook);
$writer->setOffice2003Compatibility(true);

$records = './fichier.xlsx';

$writer->save($records);
```

2.3. Afficher le tableur

Il est également possible de générer un tableur et de l'afficher plutôt que de l'enregistrer.
Pour se faire, il faut passer les entêtes à la sortie standard.

2.3.1. Excel 2007 sur le navigateur

```
include 'PHPExcel.php';
include 'PHPExcel/Writer/Excel2007.php';

$workbook = new PHPExcel;

$sheet = $workbook->getActiveSheet();
$sheet->setCellValue('A1','MaitrePylos');

$writer = new
PHPExcel_Writer_Excel2007($workbook);

 header('Content-type:
application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet');
 header('Content-
Disposition:inline;filename=Fichier.xlsx ');
 $writer->save('php://output');
```

Ici, le fichier va demander à être ouvert via votre logiciel de tableur.
Les autres possibilités sont :

Excel 5

```
header('Content-type: application/vnd.ms-excel');
```

PDF

```
header('Content-type: application/pdf');
```

CSV

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

HTML

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

3. Pour la suite...créons nos outils

Pour la suite de cet article, je me baserai sur l'affichage du tableur.
Afin de ne pas réécrire l'ensemble des codes, je vais étendre la classe PHPExcel, et commencer à créer des outils personnalisés pour la génération des fichiers.

3.1. Etendre PHPExcel

Soit la nouvelle classe suivante :

```
/**
 * Classe étendue de PHPExcel.
 * Cette classe permet la génération dynamique
 * de fichier tableur suivant différent format.
 *
 * @copyright  2008 MaitrePylos Technologies
(Formatux.be)
 * @author     Ernaelsten Gerard
 * @license    GPL
 * @version    Release: 0.1
 */

require 'PHPExcel.php';
class MaitrePylosExcel extends PHPExcel {


  public function  __construct() {
    parent::__construct();
  }

  /**
   * Méthode englobant une fonction switch pour
le choix du format de tableur.
   * @method affiche
   * @param String $format
   * @param String $nomFichier
   * @example $workbook->
affiche('Excel2007','MonFichier');
   */
  public function affiche($format = 'Excel5',
$nomFichier = 'Tableur'){

    switch($format){
      case 'Excel2007' :
        include 'PHPExcel/Writer/Excel2007.php';
        $writer = new
PHPExcel_Writer_Excel2007($this);
        $ext  = 'xlsx';
        $header =
'application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet';
        //supprime le pré-calcul
        $writer->setPreCalculateFormulas(false);
        break;
      case 'Excel2003' :
        include 'PHPExcel/Writer/Excel2007.php';
        $writer = new
PHPExcel_Writer_Excel2007($this);
        $writer->
setOffice2003Compatibility(true);
        $ext  = 'xlsx';
        $header =
'application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet';
        //supprime le pré-calcul
        $writer->setPreCalculateFormulas(false);
        break;
      case 'Excel5' :
        include 'PHPExcel/Writer/Excel5.php';
        $writer = new
PHPExcel_Writer_Excel5($this);
        $ext = 'xls';
        $header = 'application/vnd.ms-excel';
        break;
      case 'CSV' :
        include 'PHPExcel/Writer/CSV.php';
        $writer  = new
PHPExcel_Writer_CSV($this);
        $writer->setDelimiter(",");//l'opérateur
de séparation est la virgule
        $writer->setSheetIndex(0);//Une seule
feuille possible
        $ext = 'csv';
        $header = 'text/csv';
        break;
      case 'PDF' :
        include 'PHPExcel/Writer/PDF.php';
        $writer  = new
PHPExcel_Writer_PDF($this);
        $writer->setSheetIndex(0);//Une seule
feuille possible
        $ext = 'pdf';
        $header = 'application/pdf';
        break;
      case 'HTML' :
        include 'PHPExcel/Writer/HTML.php';
        $writer  = new
PHPExcel_Writer_HTML($this);
        $writer->setSheetIndex(0);//Une seule
feuille possible
```

```
        $ext = 'html';
        $header = 'text/html';
        break;

    }

    header('Content-type:'.$header);
    header('Content-
Disposition:inline;filename='.$nomFichier.'.'.
$ext);
    $writer->save('php://output');
  }

  /**
   * Méthode englobant une fonction switch pour
le choix du format de tableur.
   * @method enregistre
   * @param String $format
   * @param String $nomFichier
   * @example $workbook->
enregistre('Excel2007','MonFichier');
   */
  public function enregistre($format = 'Excel5',
$nomFichier = 'Tableur'){

    switch($format){
      case 'Excel2007' :
        include 'PHPExcel/Writer/Excel2007.php';
        $writer = new
PHPExcel_Writer_Excel2007($this);
        $ext  = 'xlsx';
        //supprime le pré-calcul
        $writer->setPreCalculateFormulas(false);
        break;
      case 'Excel2003' :
        include 'PHPExcel/Writer/Excel2007.php';
        $writer = new
PHPExcel_Writer_Excel2007($this);
        $writer->
setOffice2003Compatibility(true);
        $ext  = 'xlsx';
        //supprime le pré-calcul
        $writer->setPreCalculateFormulas(false);
        break;
      case 'Excel5' :
        include 'PHPExcel/Writer/Excel5.php';
        $writer = new
PHPExcel_Writer_Excel5($this);
        $ext = 'xls';
        break;
      case 'CSV' :
        include 'PHPExcel/Writer/CSV.php';
        $writer  = new
PHPExcel_Writer_CSV($this);
        $writer->setDelimiter(",");//l'opérateur
de séparation est la virgule
        $writer->setSheetIndex(0);//Une seule
feuille possible
        $ext = 'csv';
        break;
      case 'PDF' :
        include 'PHPExcel/Writer/PDF.php';
        $writer  = new
PHPExcel_Writer_PDF($this);
        $writer->setSheetIndex(0);//Une seule
feuille possible
        $ext = 'pdf';
        break;
      case 'HTML' :
        include 'PHPExcel/Writer/HTML.php';
        $writer  = new
PHPExcel_Writer_HTML($this);
        $writer->setSheetIndex(0);//Une seule
feuille possible
        $ext = 'html';
        break;

    }

    $writer->save($nomFichier.'.'.$ext);

  }
}
```

3.2. Petit exemple

Nous utiliserons cette classe de cette façon :

```
include 'MaitrePylosExcel.php';

$workbook = new MaitrePylosExcel();

$sheet = $workbook->getActiveSheet();
$sheet->setCellValue('A1','MaitrePylos');

$workbook-
>affiche('Excel2007','MonPremierFichier');
```

Vous prêterez attention au fait que dans l'article j'utilise la variable $sheet, qui sera le remplacement de $workbook->getActiveSheet(), dans les exemples fournis par l'application, ils utiliseront plus l'écriture $workbook->getActiveSheet()->getFont(), or je privilégie $sheet->getFont(), gardez bien cela à l'esprit.

3.3. Les formules

Pour introduire des formules dans le fichier , il suffit de passer en argument la dite formule.

```
include 'MaitrePylosExcel.php';

$workbook = new MaitrePylosExcel();

$sheet = $workbook->getActiveSheet();
$sheet->setCellValue('A1',4);
$sheet->setCellValue('A2',5);
$sheet->setCellValue('A3','=SUM(A1:A2)');

$workbook-
>affiche('Excel2007','MaPremiereFormule');
```

3.4. Multi feuilles (sheets)

Ici nous allons voir comment faire pour avoir plusieurs feuilles dans le tableur.

Le fait d'avoir plusieurs feuilles ne vous permettra pas de générer le tableur dans les formats suivants :

- PDF
- CSV
- HTML

Ces format ne permettant que la génération d'une seule feuille.

Lors de la création de votre document, une feuille est automatiquement créée, et nous en prenons possession avec la méthode getActiveSheet().

```
include 'MaitrePylosExcel.php';

$workbook = new MaitrePylosExcel();

$sheet = $workbook->getActiveSheet();

$workbook-
>affiche('Excel2007','MaPremiereFormule');
```

Pour donner un nom à cette feuille, on utilise la méthode setTitle()

```
 $sheet->setTitle('feuille 1');
```

Pour créer de nouvelles feuilles, nous utiliserons la méthode createSheet() de la classe PHPExcel, qui elle même instancie la classe PHPExcel_WoorkSheet.

```
/*****************************************
         *
         * Création de la deuxième feuille
         *********************************/
        //création de la nouvelle feuille
        $sheet2 = $workbook->createSheet();
        //nom de la feuille
        $sheet2->setTitle('feuille 2');



       /************************************
       *
       * Création de la troisème feuille
       ************************************/
        $sheet3 = $workbook->createSheet();
        $sheet3->setTitle('feuille 3');


        /************************************
       *
       * Création de la quatrième feuille
       ************************************
        $sheet4 = $workbook->createSheet();
        $sheet4->setTitle('feuille 4');
```

4. Mise en forme

PHPExcel nous offre 8 classes pour mettre en forme nos cellules :

- PHPExcel_Style
- PHPExcel_Style_Fill
- PHPExcel_Style_Font
- PHPExcel_Style_Border
- PHPExcel_Style_Borders
- PHPExcel_Style_Alignment
- PHPExcel_Style_NumberFormat
- PHPExcel_Style_Protection

Nous n'allons pas voir l'ensemble des classes, mais nous verrons les 2-3 approches qu'il est possible de faire, ensuite je vous renvoie vers la documentation officielle, pour explorer plus en avant l'ensemble des classes.

4.1. La classe de style "Font"

Dans la classe PHPExcel_Style_Font, les méthodes importantes sont les suivantes :

- setName()
- setBold()
- setItalic()
- setUnderline()
- setStriketrough()
- getColor()
- setSize()
- setSuperScript()
- setSubScript()

La plupart de celles-ci sont assez explicites.

4.1.1. MaitrePylos en vert...

Nous allons mettre en forme une cellule en fonction des critères suivants :

- En gras
- De taille 12
- De police 'Arial'
- De couleur Verte

Pour ce faire il suffit d'écrire le code suivant :

```
$workbook = new MaitrePylosExcel();

$sheet = $workbook->getActiveSheet();

$styleA1 = $sheet->getStyle('A1');
$styleFont = $styleA1->getFont();
$styleFont->setBold(true);
$styleFont->setSize(12);
$styleFont->setName('Arial');
$styleFont->getColor()-
>setARGB(PHPExcel_Style_Color::COLOR_GREEN);
$sheet->setCellValue('A1','MaitrePylos');

$workbook-
>affiche('Excel2007','MonPremierFichier');
```

Nous allons passer en revue ce code bien comprendre ce qu'il se passe.

```
$styleA1 = $sheet->getStyle('A1');
```

getStyle() est une méthode de la classe PHPExcel_Worksheet qui instancie la classe PHPExcel_Style (que nous verrons plus tard), on lui passe en paramètre la cellule visée.

```
$styleFont = $styleA1->getFont();
```

getFont() est donc une méthode de la classe PHPExcel_Style, qui instancie la classe PHPExcel_Style_Font.
A partir de cette instanciation nous pouvons utiliser les méthodes de PHPExcel_Style_Font.

```
$styleFont->setBold(true);//passage à true de
façon à activer la graisse.
$styleFont->setSize(12);//taille de la police
$styleFont->setName('Arial');//nom de la police
$styleFont->getColor()-
>setARGB(PHPExcel_Style_Color::COLOR_GREEN);
```

Pour la couleur, c'est un peu plus compliqué, il faut repasser par une nouvelle classe PHPExcel_Style_Color
Celle-ci défini un certain nombre de constantes que voici :

Constante	Valeur
COLOR_BLACK	'FF000000'
COLOR_WHITE	'FFFFFFFF'
COLOR_RED	'FFFF0000'
COLOR_DARKRED	'FF800000'
COLOR_BLUE	'FF0000FF'
COLOR_DARKBLUE	'FF000080'
COLOR_GREEN	'FF00FF00'
COLOR_DARKGREEN	'FF008000'
COLOR_YELLOW	'FFFFFF00'
COLOR_DARKYELLOW	'FF808000'

Ainsi que deux méthodes : setARGB(), que vous connaissez déjà (voir exemple ci-dessus) et qui prend en paramètre une des constantes.
setRGB(), qui, elle prend en paramètre le code HTML de la couleur :

```
$styleFont->getColor()->setRGB('FF00FF00');
```

4.1.2. applyFromArray

Il y a moyen d'alléger ce code. En effet, la classe PHPExcel_Style_Font propose une méthode applyFromArray() qui prend un tableau en paramètre afin de ne pas utiliser les setter et getter de la classe et de rendre le code plus lisible.
Les mots clés de ce tableau sont :

- name
- bold
- italic
- underline
- strike
- color
- size
- superScript
- subScript

Les noms sont fort proches des méthodes, et pour cause, elles utiliseront celles-ci, pour définir notre style de cellule.
Notre nouveau code sera donc :

```
$styleA1 = $sheet->getStyle('A1');
$styleFont = $styleA1->getFont()
->applyFromArray(array(
    'bold'=>true,
    'size'=>12,
    'name'=>Arial,
    'color'=>array(
        'rgb'=>'FF00FF00')));
```

Ce simple tableau remplace tous les appels aux méthodes de PHPExcel_Style_Font, puisqu'il le fait lui-même.
Vous remarquerez que color demande un nouveau tableau, parce que la classe PHPExcel_Style_Color à également une méthode applyFromArray, avec soit rgb ou argb.

4.2. PHPExcel_Style

PHPExcel_Style est la classe qui nous permet de travailler avec les autres classes de 'style'.
Quand nous appelons la méthode getStyle(), en fait nous instancions la classe PHPExcel_Style, celle-ci dispose de méthodes nous permettant d'instancier les autres classes de styles,
les méthodes sont :

- getFill()
- getFont()
- getBorders()
- getAlignment()
- getNumberFormat()
- getProtection()

Chaque méthode instancie la classe par rapport à son nom :
- getFont() instancie PHPExcel_Style_Font.
- getAlignment() instancie PHPExcel_Style_Alignment.
...

4.2.1. applyFromArray de PHPExcel_Style

PHPExcel_Style dispose également d'une méthode applyFromArray contenant le tableau suivant :

- fill
- font
- borders
- alignment
- numberformat
- protection

Notre code deviendra dès lors :

```
$styleA1 = $sheet->getStyle('A1');
$styleA1->applyFromArray(array(
    'font'=>array(
        'bold'=>true,
        'size'=>12,
        'name'=>Arial,
        'color'=>array(
            'rgb'=>'FF00FF00'))
    ));
```

Retrouvez la suite de l'article de Gérard Ernaelsten en ligne : Lien21

Comprendre l'amélioration progressive

Cet article est la traduction de Understanding Progressive Enhancement disponible en anglais ici (Lien22).
Translated with the permission of A List Apart Magazine and the author.
Retrouvez toutes les traductions disponibles de A List Apart Magazine sur alistapart.developpez.com.

Depuis 1994, la communauté du développement web a battu le tambour de la dégradation élégante (Lien23). Transposition du monde de l'ingénierie, le concept était, à sa base, de réserver un repas complet aux navigateurs les plus récents et les plus performants tout en laissant quelques restes aux tristes sires assez malchanceux pour utiliser Netscape 4. Cela a fonctionné, bien sûr, mais ça ne correspondait pas vraiment à la vision originelle de Tim Berners-Lee d'un web universellement accessible.
Une dizaine d'années plus tard, des personnes intelligentes se sont interrogées sur la dégradation élégante et l'ont trouvée insuffisante à de nombreux niveaux. Préoccupées par la disponibilité du contenu, l'accessibilité globale et les capacités des navigateurs mobiles, elles ont cherché une nouvelle méthode d'approche du développement web - une méthode qui se concentre sur le contenu et ne se contente pas de faire semblant avec les anciens outils.
Au SXSW en 2003, Steve Champeon et Nick Finck ont présenté une conférence intitulée "Conception Web Inclusive pour le Futur" (Lien24). Ils ont alors dévoilé un modèle pour cette nouvelle méthode d'approche du développement web. Steve lui a aussi donné un nom : l'amélioration progressive (Lien25).

1. Il y a une (subtile) différence

Au cas où vous vous creuseriez la tête en essayant de trouver en quoi la dégradation élégante et l'amélioration progressive diffèrent, je vous dirais ceci : c'est une question de perspective. La dégradation élégante et l'amélioration progressive considèrent toutes deux le fonctionnement d'un site dans de nombreux navigateurs, sur de nombreux systèmes. La clé est leur point d'intérêt et la façon dont il affecte l'organisation du développement.

1.1. La perspective de la dégradation élégante

La dégradation élégante se concentre sur la création de sites web pour les navigateurs les plus avancés / évolués. Tester sur des navigateurs considérés "plus anciens" ou moins évolués est généralement fait dans le dernier quart du cycle de développement et est souvent restreint à la précédente version des principaux navigateurs (IE, Mozilla, etc.).

Dans le cadre de ce paradigme, le résultat donné par les navigateurs plus anciens est censé être pauvre, mais suffisant. De petites corrections peuvent être faites pour s'adapter à un navigateur particulier. Comme ils ne sont pas le centre d'intérêt, peu d'attentions sont portées au-delà de la correction des erreurs les plus flagrantes.

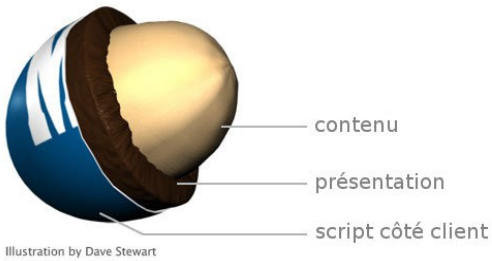
1.2. La perspective de l'amélioration progressive

L'amélioration progressive se concentre sur le contenu. Notez la différence : *je n'ai même pas mentionné les navigateurs*.

A l'origine, nous créons des sites web pour leur contenu. Certains sites le propagent, d'autres le collectent, d'autres en demandent, d'autres en manipulent et même certains font tout ça, mais tous en ont besoin. C'est ce qui fait de l'amélioration progressive un paradigme plus approprié. C'est pourquoi Yahoo! l'a rapidement adopté et l'utilise pour créer sa stratégie de support noté des navigateurs (Lien26).

2. Mais alors, comment ça marche ?

Se faire à la mentalité de l'amélioration progressive est assez simple : pensez simplement contenu. Le contenu constitue la base solide sur laquelle vous appliquez votre style et votre interactivité. Si vous êtes fan de bonbons, représentez-vous l'amélioration progressive comme une cacahuète M&M's :



Les couches de chocolat de l'amélioration progressive

Commencez avec votre cacahuète de contenu, balisée avec du (X)HTML riche et sémantique. Habillez ce contenu d'une couche de CSS riche et crémeux. Finalement, ajoutez du JavaScript en coque de sucre dur pour faire un bonbon merveilleusement savoureux (et éviter qu'il fonde

dans vos mains).

Si vous êtes familiarisés avec le mantra du mouvement des standards Web - séparation, séparation, séparation - c'est parfaitement logique. Le développement basé sur les standards Web a souvent été comparé à un gâteau fourré (Lien27) (ou, si vous voulez vraiment frimer, un trifle (Lien28)).

Je préfère l'analogie à la cacahuète M&M's, parce qu'avec, les couches enrobent complètement le contenu, de la même façon que les styles et les scripts enveloppent notre contenu.

Si vous suivez mon analogie culinaire un peu plus longtemps (j'espère que je ne vous donne pas faim), je vous expliquerai pourquoi cette approche est meilleure et comment les couches communiquent dans ce paradigme.

2.1. La cacahuète

Certaines personnes aiment les cacahuètes ; en fait, certaines personnes préfèrent les cacahuètes aux M&M's. De la même façon, certains (et des éléments comme les moteurs de recherche) veulent uniquement le contenu.

Ensuite, il y a des personnes qui ne supportent pas le chocolat et les couches de sucre sur la cacahuète (les diabétiques, par exemple). Comme eux, certaines personnes sur des appareils mobiles ou de vieux navigateurs peuvent être incapables de voir vos belles présentations ou interagir avec vos astucieuses interfaces gérées par AJAX.

S'assurer que vos balises expriment le meilleur niveau de détail sur le contenu qu'elles entourent est essentiel pour offrir une expérience de base à ces utilisateurs.

2.2. L'enrobage de chocolat

Après, vous pouvez délicatement faire baigner votre contenu dans un bain chaud de CSS, mais avant de sauter sur la gangue de sucre dur, il y a certaines considérations supplémentaires.

Il y a des gens qui aiment que le chocolat recouvre les cacahuètes. Ces gens sont comme le tiers moyen des utilisateurs qui ont un certain niveau de support CSS, mais ne peuvent avoir de support décent pour JavaScript, ou ils peuvent travailler dans une entreprise où les gens de la sécurité IT sont plus qu'un peu phobiques envers JavaScript. Pour eux, JavaScript peut être complètement désactivé.

Qu'elles préfèrent le Goober (Lien29) [NdT : cacahuète enrobée de chocolat, mais sans couche de sucre], ou soient limitées au Goober (Lien29), ces personnes méritent d'être accueillies. Il existe plusieurs techniques d'amélioration progressive permettant d'appliquer des styles à votre contenu et ce sera le sujet du deuxième article de cette série.

2.3. La coque de sucre dur

Pour finir, vous pouvez introduire JavaScript dans le mélange. Avec les riches possibilités d'interaction qu'il propose, ainsi que sa capacité à manipuler et interagir avec les couches de contenu et de présentation, JavaScript est réellement l'ingrédient qui peut pousser un site dans son ensemble vers une "expérience".

Je ne suis pas vraiment sûr de la façon dont la coque de sucre dur est ajoutée à un M&M's (je suppose qu'il s'agit d'un bain supplémentaire), mais ajouter des fonctionnalités et de l'interactivité basées sur JavaScript à vos sites web est un jeu d'enfant quand vous pensez à l'amélioration progressive. Et, de la même façon que les M&M's sont disponibles dans une variété de couleurs, l'expérience JavaScript peut varier en fonction des possibilités du navigateur ou du système qui essaie de l'utiliser.

Comme vous le savez certainement, ce type de développement est appelé JavaScript non-intrusif. Je traiterai ces techniques et pratiques dans le troisième et dernier article de cette série.

3. Tout rassembler

Développer avec l'amélioration progressive est en fait assez simple une fois que vous avez compris le concept et commencé à le mettre en pratique ; peut-être même plus simple que de faire du sucre. Les deux prochains articles de cette série vous aideront à aiguiser vos compétences en amélioration progressive avec CSS (Lien30) et JavaScript (Lien31) et vous montreront comment cette philosophie se traduit en code.

Retrouvez l'article de Aaron Gustafson traduit par Sylvain Jorge Do Marco en ligne : Lien32

Personnalisez vos boîtes de message (X)HTML

Cet article est la traduction de CSS Message Boxes for different message types disponible en anglais ici (Lien33). Dans cet article, nous allons voir comment faire des boîtes de message personnalisées en fonction du type de message.

1. Introduction

Pouvez-vous y croire : Récemment, je suis allé à la banque vérifier l'état de mon compte. L'employé entre mes données personnelles et envoie une requête. L'application Web répond en affichant une boîte de message jaune avec un point d'exclamation indiquant que la demande était en cours. Il vérifia plusieurs autres fois, mais n'a pas fait attention qu'à un moment, le message se changea en "Compte valide". La boîte de message n'avait pas changé. Il continua d'essayer et s'aperçu enfin que la demande avait réussi.

Je ne sais pas ce qu'il y avait dans la tête des développeurs, mais ils ne pensaient sûrement pas aux utilisateurs. Ce pauvre employé était réellement embêté. Je n'ose pas imaginer à quoi doit ressembler le reste de l'application.

Pour éviter cela, différents types de boîtes de message devraient être affichés différemment. Je pense que chaque application Web devrait prévoir quatre types de messages : information, opération réussie, alerte et erreur. Chaque type de message devrait être affiché avec une couleur et une icône spécifique. Il existe aussi un type particulier de message que sont les validations.

Je vais vous montrer un remake des boîtes de message que j'ai utilisées pour mon dernier projet. Je les ai légèrement modifiées pour qu'ils soient plus simples dans cet exemple. Vous verrez aussi comment donner un style particulier de façon similaire pour vos messages de validation.

Voyons maintenant les différents types de messages.

2. Les messages d'information

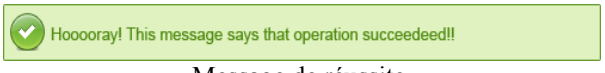
Le but de ce type de message est d'informer l'utilisateur de quelque chose d'important. Cela peut être présenté en bleu car les gens associent souvent cette couleur à une information, indépendamment du contenu. Cela peut correspondre à n'importe quelle information concernant les actions de l'utilisateur.

Message d'information

Par exemple, les messages d'information peuvent afficher une aide ou un conseil concernant l'action en cours.

3. Messages de réussite

Les messages de réussite devraient être affichés après le succès d'une action de l'utilisateur. J'entends par là une opération complètement terminée, pas partiellement terminée et sans erreur jusque là. Le message pourrait dire par exemple : "Votre profil a été enregistré avec succès et un mail de confirmation vous a été adressé à l'adresse que vous avez indiquée". Ce qui signifie que toutes les étapes ont été effectuées avec succès (enregistrement du profil et envoi de mail).



Message de réussite

Je sais que beaucoup de développeurs considèrent ce genre de message comme un message d'information, mais je préfère les afficher avec leur propre charte - couleur verte et icône "check".

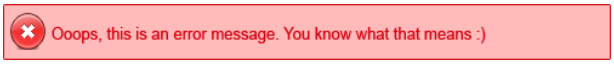
4. Messages d'alerte

Les messages d'alerte devraient être affichés lorsqu'une opération n'a pas pu se terminer complètement. Par exemple "Votre profil a bien été enregistré, mais nous n'avons pas pu envoyer de mail à votre adresse" ou "Si vous ne créez pas votre profil, vous ne pourrez pas chercher d'emploi". Habituellement, ces messages sont affichés en jaune avec un point d'exclamation.

Message d'alerte

5. Messages d'erreur

Les messages d'erreur devraient être affichés lorsque l'opération n'a pas pu se réaliser du tout. Par exemple : "Votre profil n'a pas pu être créé." La couleur rouge est appropriée pour ces messages car les gens l'associent à tous les types d'alerte.



Message d'erreur

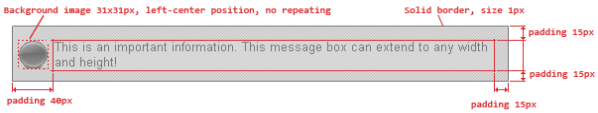
6. Procédé de design

Maintenant que nous savons comment présenter les messages, voyons comment les créer en CSS. Faisons un tour rapide des procédés de design.

Je vais faire cela le plus simplement possible. Le but est d'obtenir une seule *div* avec une classe unique. Le code HTML ressemblera donc à ça :

```
<div class="info">Message d'information</div>
<div class="success">Message de réussite</div>
<div class="warning">Message d'alerte</div>
<div class="error">Message d'erreur</div>
```

La classe CSS va ajouter une image de fond à la *div* sur la gauche. Nous mettrons aussi un padding pour qu'il y ai suffisamment d'espace autour du texte. Notez que le padding gauche doit être plus important pour que le texte ne chevauche pas l'image.



Modèle

Et voici les quatre classes CSS (cinq avec les messages de validation) correspondant aux messages :

```
body{
        font-family:Arial, Helvetica, sans-serif;
        font-size:13px;
}
.info, .success, .warning, .error, .validation {
        border: 1px solid;
        margin: 10px 0px;
        padding:15px 10px 15px 50px;
        background-repeat: no-repeat;
        background-position: 10px center;
}
.info {
        color: #00529B;
        background-color: #BDE5F8;
        background-image: url('info.png');
}
.success {
        color: #4F8A10;
        background-color: #DFF2BF;
        background-image:url('success.png');
```

```
}
.warning {
        color: #9F6000;
        background-color: #FEEFB3;
        background-image: url('warning.png');
}
.error {
        color: #D8000C;
        background-color: #FFBABA;
        background-image: url('error.png');
}
```

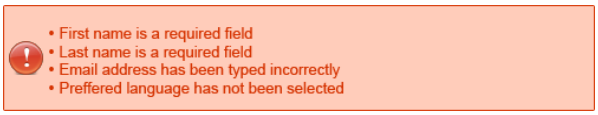
Les icônes utilisées dans cet exemple proviennent de Knob Toolbar icons (Lien34).

Voir un exemple (Lien35).

7. Messages de validation

J'ai remarqué que de nombreux développeurs ne font pas de distinction entre un message de validation et les autres (comme les erreurs ou alertes). J'ai souvent vu des messages de validation affichés comme des erreurs, créant une confusion chez l'utilisateur.

La validation concerne tout ce que l'utilisateur a entré et devrait être traité comme tel. ASP.NET possède des contrôles qui permettent de vérifier toutes les données entrées par l'utilisateur. Le but de la validation est de forcer l'utilisateur à entrer toutes les informations obligatoires dans le bon format. De ce fait, il doit être évident que le formulaire ne sera pas soumis si les données ne sont pas conformes. C'est pour cela que j'aime afficher les messages de validation dans un rouge un peu moins intense et avec un point d'exclamation.



Message de validation

La classe CSS pour les messages de validation est similaire aux autres messages (notez que certains attributs ont été définis dans le code précédent) :

```
.validation {
    color: #D63301;
    background-color: #FFCCBA;
    background-image: url('validation.png');
}
```

8. Conclusion

Les messages sont un élément important pour l'utilisateur et sont souvent négligés. Il existe beaucoup d'articles présentant de très belles boîtes de message, mais ce n'est pas uniquement une question d'esthétique. Ils doivent apporter une information avec un sens, que ce soit visuellement ou sémantiquement.

Voir aussi cet article (Lien36) pour découvrir comment animer ces messages avec jQuery.

Retrouvez l'article de Janko Jovanovic traduit par Didier Mouronval en ligne : Lien37

Espaces de noms en JavaScript

Cet article s'adresse à une population de développeurs déjà familière avec la problématique du développement Web et le langage JavaScript. Dans cet article, vous apprendrez comment packager vos bibliothèques en simulant en JavaScript ce que l'on appelle un espace de noms ("namespace" en anglais).

1. Qu'est-ce qu'un espace de noms

Faisons un bref rappel de ce qu'est un espace de noms (ou une courte introduction pour ceux qui ne les connaissent pas).

1.1. Problématique

Lorsque vous développez, seul ou en équipe, il peut arriver que vous vous retrouviez dans plusieurs codes sources avec des variables, des fonctions, des classes, etc. portant le même nom mais effectuant des tâches différentes car prévues pour des contextes différents. Si vous devez faire collaborer ces codes au sein d'un même programme, il y aura alors conflit de noms. Avec un langage compilé le compilateur vous préviendra de ce conflit, mais avec un langage interprété comme le langage JavaScript vous passerez à côté et ce que vous manipulerez sera la dernière définition de vos variables, fonctions, etc. qui aura écrasé les précédentes.

1.2. Comment s'en sortir

Sauvez-nous docteur! Et bien voilà toute l'utilité des espaces de noms. Cela sert à regrouper dans un même espace vos sources et à les qualifier de manière unique (si vous ne vous amusez pas à créer deux espaces de noms similaires évidemment).

1.3. Comment choisir de mettre telle source dans un espace et telle autre dans un autre

Il est préférable de regrouper sous un même espace de noms les fonctionnalités qui agissent de conserve dans un même contexte. Par exemple, il n'est pas pertinent de mettre dans le même espace de noms des fonctionnalités pour des traitements mathématiques et pour des conteneurs. Dans ce cas, il est plus judicieux de créer 2 espaces de noms : un pour ce qui se rapporte aux nombres ("Math" par exemple) et aux chaînes de caractères ("String" par exemple). Dans ces deux espaces de noms vous pouvez avoir une fonction "Add". Dans le cas de "Math" elle peut additionner deux nombres, dans le cas de "String" elle peut concaténer le contenu d'une chaîne à une autre (addition de chaînes), deux actions complètement différentes qui pourtant portent assez logiquement le même nom.

J'espère que cela vous a convaincu de l'utilité des espaces de noms. Voyons maintenant comment les mettre en oeuvre en JavaScript.

2. Création d'un espace de noms en JavaScript

Vais-je vous choquer et vous faire vous demander quel est le but de cet article si je vous dis que les espaces de noms n'existent pas en JavaScript : Et non, ça n'existe pas en tant que tel ! Mais il est possible de les simuler.

En JavaScript, vous avez la possibilité de créer des objets et de leur adjoindre des attributs (ou propriétés) comme bon vous semble, aussi que sera un espace en JavaScript ? Un objet dont les propriétés seront les fonctionnalités que vous cherchez à regrouper dans un même domaine.

2.1. Les bases

Créons donc un espace de noms, i.e. un objet, myNamespace :

Création d'un espace de noms
```
var myNamespace = {} ;
```

Ajoutons lui quelques fonctionnalités :

Ajout de fonctionnalité à un espace de noms
```
myNamespace.var1 = "NF";
myNamespace.var2 = 28;
myNamespace.displayVar1 = function()
{ alert(this.var1) ; } ;
myNamespace.displayVar2 = function()
{ alert(this.var2) ; } ;
myNamespace.createObject = function(){
  return new function()
  {
    this.name = "no name";
    this.what = "object created by
myNamespace.createObject";
    this.version = "v1.0";
    this.display = function() {
      alert(this.name+ " ("+this.what+") - "+
this.version);
    }
  }
};
```

Et testons notre création :

Utilisation des fonctionnalités d'un espace de noms
```
alert(myNamespace.var1);    // NF
myNamespace.var1 = "NF is the author";
myNamespace.displayVar1(); // NF is the author

alert(myNamespace.var2);    // 28
myNamespace.var2++;
```

```
myNamespace.displayVar2(); // 29

var o = myNamespace.createObject();
o.version = "v2.0";
o.display();
// no name (objet créé par
myNamespace.createObject) - v2.0
```

Quelques explications s'imposent.

Tout d'abord nous avons créé un objet appelé myNamespace dont nous supposeront que le nom est unique (à vous de faire en sorte que ce soit le cas).

Puis nous lui avons adjoint quelques propriétés qui constituent les fonctionnalités de notre bibliothèque :

- Une variable var1 contenant une chaîne de caractères
- Une variable var 2 contenant un nombre
- Une fonction qui affiche var1
- Une fonction qui affiche var2
- Une fonction qui crée un objet possédant 4 propriétés (3 variables et une fonction d'affichage de ses variables)

Enfin nous utilisons les fonctionnalités de notre bibliothèque :

- Affiche var1
- Modifie la valeur de var1
- Affiche var1 (nouvelle valeur)
- Affiche var2
- Modifie la valeur de var2
- Affiche var2 (nouvelle valeur)
- Crée un objet o
- Modifie la valeur de o.version
- Affiche les valeurs des attributs de o

A ce stade, vous avez packagées vos fonctionnalités (i.e. votre bibliothèque), vous avez résolu d'éventuels conflits de noms en regroupant ces fonctionnalités dans un même espace (ou domaine) et si le choix de l'espace de noms est pertinent, on reconnaît tout de suite à quelle utilisation se destine votre bibliothèque.

2.2. Notation JSON

Peut-être que le code précédent n'est pas lisible pour vous, pas assez intuitive. Peut-être êtes-vous habitué à une autre syntaxe du fait de votre expérience passée et vous abordez celle-ci avec difficulté. Sachez qu'il ne s'agit pas là de l'unique syntaxe possible.

Vous pouvez par exemple utiliser la notation JSON :

Notation JSON

```
var myNamespace = {
  var1 : "NF",
  var2 : 28,
  displayVar1 : function()
{ alert(this.var1) ; },
  displayVar2 : function()
{ alert(this.var2) ; },
  createObject : function(){
    return {
      name : "no name",
      what : "objet créé par
myNamespace.createObject",
      version : "v1.0",
      display : function() {
        alert(this.name+ " ("+this.what+") - "+
this.version);
      }
    }
  }
};
```

Cette notation est strictement équivalente à la précédente, plus élégante car plus lisible et montrant tout de suite que les fonctionnalités de votre bibliothèques appartiennent à un même ensemble myNamespace.

Cependant, rien ne vous empêche de mixer les 2 notations. Par exemple, si vous avez implémenté votre bibliothèque dans un fichier et que dans un second vous souhaitez en ajouter de nouvelles. Dans votre premier fichier écrire le code précédent (Listing 3 4) et dans le second :

Extension d'un namespace

```
myNamespace.newFunction = function() {
  alert("new function added") ;
};
```

2.3. Existence et unicité d'un espace de noms

Si vous êtes amené à séparer dans plusieurs fichiers l'implémentation de votre espace de noms comme dans le section précédente, je vous conseille de tester son existence au début de chaque fichier d'extension, car rien ne vous garanti que les utilisateurs de votre bibliothèque inclueront tous les scripts nécessaires dans leurs pages web, ou qu'ils les déclarent dans le bon ordre.

Test de l'existence du nom de la variable

```
if (typeof myNamespace == "undefined")
{
  alert("myNamespace is not defined. Include
myNamespace.js before this script");
}
else
{
  myNamespace.newFunction = function() {
    alert("new function added") ;
  };
}
```

Depuis le début de cet article, je vous dis que les espaces de noms nous permettent d'éviter les conflits de noms. C'est bien joli me direz-vous, mais si l'on package notre code au sein d'espaces de noms qu'est-ce qui empêche :

- qu'on se retrouve avec plusieurs espaces de noms portant le même nom ?
- qu'un objet ou une variable ait le même nom que notre espace de noms ?

Réponse : rien, retour à la case départ.

En fait pas tout à fait, car déjà nous avons évité qu'au sein de vos propres développements vous ayez des conflits de noms et en travail collaboratif nous avons limité la possibilité de conflits aux noms des espaces. C'est déjà pas mal. Comment atteindre l'étape ultime et éviter tout conflit

entre votre espace et un objet ou une variable ? Impossible. Vous ne pouvez garantir que ceux qui utiliseront votre espace de noms feront attention à ne pas créer autre chose portant le même nom.

Cependant vous pouvez mettre en place un système pour détecter si à la fin du chargement d'une page web utilisant votre espace celui-ci n'a pas été écrasé par une autre variable de même nom. Vous pouvez aussi étoffer le test du listing précédent pour non seulement tester l'existence du nom de votre espace, mais aussi vérifier qu'il s'agit toujours de votre espace.

Pour cela ajoutons une propriété author à notre espace dans laquelle vous mettez votre nom. Cette propriété nous sert à identifier l'espace de noms :

Identifier l'espace de noms

```
var myNamespace = {
  author : "Nourdine FALOLA",
  var1 : "NF",
  var2 : 28,
  displayVar1 : function()
{ alert(this.var1) ; },
  displayVar2 : function()
{ alert(this.var2) ; },
  createObject : function(){
    return {
          name : "no name",
          what : "objet créé par
myNamespace.createObject",
          version : "v1.0",
          display : function() {
            alert(this.name+ " ("+this.what+") -
"+ this.version);
          }
    }
  }
};
```

Et structurons chaque fichier d'extension de notre espace de la façon suivante :

Test de l'existence et de l'unicité de l'espace de noms

```
(function()
{
  var f = function()
  {
    if (typeof myNamespace == "undefined")
    {
      alert("the namespace myNamespace was
deleted.");
      return false;
    }
    else if (typeof myNamespace.author ==
"undefined" ||
             myNamespace.author != "Nourdine
FALOLA")
    {
      alert("the namespace myNamespace was
altered.");
      return false;
    }

    return true;
  }

  if (window.addEventListener) // non-IE browser
  {
    window.addEventListener("load",f,false);
  }
  else if (window.attachEvent) // IE browser
  {
    window.attachEvent("onload",f);
  }

  if (!f())
    return;

  // put your extension code here

})(); // run the anonymous function
```

Que réalise la fonction anonyme dont la définition représente tout notre fichier d'extension et qui est exécutée dans la foulée :

- elle définie une fonction f vérifiant l'existence de la variable portant le nom de notre espace, l'existence d'une propriété author contenant le nom de l'auteur de la bibliothèque.
- elle attache un événement à la fin du chargement de la page appelant ce code (exécution de la fonction f)
- elle teste l'exécution de f renvoie bien true et stoppe le script si false
- elle étend l'espace de noms par ajout de nouvelles propriétés

Détaillons chaque étape.

D'abord tout le code de votre fichier d'extension est encapsulé dans une fonction anonyme exécutée dès la fin de sa définition grâce à la syntaxe suivante :

Syntaxe de l'exécution d'une fonction anonyme

```
(function() {
// code
})() ;
```

Une fonction f est définie. Cette fonction réalise plusieurs tests pour vérifier que la variable représentant l'espace de nom existe et qu'il s'agit bien de celle que vous avez créée dans le fichier initial (par vérification de l'existence de la propriété author et de sa valeur), si c'est le cas elle renvoie true, sinon false.

Cette fonction est ensuite attachée à l'événement onload de la page, c'est-à-dire qu'elle sera exécutée dès la fin du chargement de la page. Pourquoi à la fin du chargement de la page ? Parce que si un utilisateur écrase votre espace, il y a de fortes chances que ce soit avant le chargement complet de la page.

Puis le retour de cette fonction est testé afin de voir tout de suite si on manipule bien la bonne variable myNamespace. Si oui, on passe à la définition des nouvelles propriétés qui vont étendre notre espace de noms, si non, la fonction s'arrête là et retourne.

Comprenez bien que si la variable n'est pas celle que vous croyez, un message d'alerte vous le dira :).

2.4. Les sous-espaces de noms

Maintenant vous savez regrouper vos sources dans un domaine et créer différents domaines pour des contextes spécifiques. C'est pas mal, mais vous pouvez encore améliorer la donne en ajoutant plusieurs niveaux de nommage. Pourquoi plusieurs niveaux de nommage ?

Tout d'abord si vous avez créé différents espaces de noms, c'est pour éviter les conflits de noms et regrouper vos fonctionnalités suivant des contextes particuliers. Mais de la même façon qu'il pouvait être pertinent d'avoir deux fonctionnalités portant le même nom, il en est de même pour les espaces de noms (et là vous commencez à avoir mal à la tête).

Ensuite, il serait bien pratique de n'avoir qu'une seule entrée pour identifier les fonctionnalités provenant de votre (vos) bibliothèque(s) et structurer suivant une hiérarchie pertinente vos fonctionnalités. En plus, n'avoir qu'une entrée vous permet de n'avoir à vous soucier que d'une possibilité de conflit de nom entre votre espace de noms global et une variable lambda.

Avec ce que vous avez appris précédemment, vous avez tous les outils pour agrémenter votre espace de noms principal de sous-espaces. Comme vous l'avez vu, un espace de noms en JavaScript est un objet dont les propriétés sont les fonctionnalités de votre bibliothèque. De la même façon, vous pouvez créer une propriété qui aura elle-même comme propriétés d'autres fonctionnalités, il s'agit alors d'un sous-espace de noms, un espace dans l'espace.

Ajoutons à notre espace de nom précédent deux sous-espaces et un sous-espace à l'un d'entre eux. Nous aurons alors 3 niveaux de profondeur.

Ajouter des sous-espaces de noms

```
myNamespace.subNamespace1 = {
  displayMsg : function(){
    alert("you're in subNamespace1");
  }
};

myNamespace.subNamespace2 = {
  displayMsg : function(){
    alert("you're in subNamespace2");
  }
};

myNamespace.subNamespace1.subSubNamespace = {
  displayMsg : function(){
    alert("you're in subSubNamespace");
  }
};
```

Testons les extensions de notre espace de noms global :

Utiliser les sous-espaces de noms

```
myNamespace.subNamespace1.displayMsg();
myNamespace.subNamespace2.displayMsg();
myNamespace.subNamespace1.subSubNamespace.displayMsg();

// you're in subNamespace1
// you're in subNamespace2
// you're in subSubNamespace
```

Vous constatez qu'au premier niveau vous avez les fonctionnalités de l'espace de noms myNamespace (var1, var2, displayVar1, displayVar2, createObject), au second niveau les fonctionnalités de subNamespace1 et subNamespace2 et au troisième niveau les fonctionnalités de subSubNamespace !

2.5. Les limites des espaces de noms en JavaScript

Jusque là nous avons été super optimistes sur notre conception d'un espace de noms en JavaScript. Pourtant, j'ai dit au début de cet article que la notion d'espace de noms n'existe pas dans ce langage et que ce que nous pouvions faire, au mieux, c'est l'émuler. Tout ceci implique certaines limitations.

Le JavaScript est un langage interprété. Aucune vérification n'est faite à la compilation puisqu'il n'y a pas de compilation. Donc si un problème survient c'est à l'exécution, ce qui est très embêtant pour l'utilisateur (et pour vous en tant que concepteur). C'est pourquoi nous avons fait ce que nous pouvions pour détecter en amont les problèmes possibles et prévenir par une alerte le cas échéant et en stoppant les traitements voués à l'échec.

Pourtant nous ne sommes pas capable de prévenir tous les problèmes. La principale limitation est que nous ne pouvons pas nous assurer de la bonne utilisation de notre bibliothèque par des tiers. Entre autres ils peuvent :

- Ecraser tout ou partie des fonctionnalités de notre bibliothèque en les redéfinissant (vil gredin !).

  Si cette fonction était utilisée quelque part en interne dans votre bibliothèque autant dire que ça ne marche plus.

Ecrasement d'une fonctionnalité dans le code client

```
// Script utilisateur
myNamespace.displayVar1 = null;
```

- Ecraser nos tests de validité sur le chargement de la page en affectant un callback sur l'événement onload façon old school (traditional binding).

Ecrasement du test de validité de l'espace de noms dans le code client

```
// Script utilisateur après l'inclusion du Listing 3 8
window.onload = function()
{ alert("gotcha !!") } ;
```

- Ecraser notre espace de noms APRES que l'on ait testé sa validité en ajoutant un autre callback à la suite du notre (avec addEventListener ou attachEvent suivant le navigateur). Auquel cas il n'y a pas d'alerte pour le prévenir du méfait.

Contournement du test de validité de l'espace de noms dans le code client

```
// Script utilisateur après l'inclusion du Listing 3 8
var f = function()
{
  myNamespace = null;
  alert("gotcha again !!");
}
```

```
if (window.addEventListener) // non-IE browser
{
  window.addEventListener("load",f,false);
}
else if (window.attachEvent) // IE browser
{
  window.attachEvent("onload",f);
}
```

- Utiliser des raccourcis pour ne pas avoir à écrire le nom de l'espace ou du sous-espace de noms, de la classe, de la variable, de la fonction en entier. De ce fait nous ne pouvons garantir que sa nouvelle variable ne sera pas écrasée par une autre qu'il utilisera ailleurs.

  Ce problème s'apparente à celui que l'on peut rencontrer dans des langages compilés comme le C++ où l'on utilise l'instruction using pour ne pas avoir à préfixer les fonctionnalités par leur espace de noms, ce qui peut induire des ambiguïtés. Donc à utiliser avec d'extrêmes précautions, voire à proscrire.

Utilisation de raccourcis d'écriture dans le code client

```
// Script utilisateur
var raccourci = myNamespace.displayVar1;
raccourci() ; // affiche la valeur de
myNamespace.var1
[...]
raccourci = myNamespace.subNamespace1.displayMsg;
raccourci() ; // affiche "you're in
subNamespace1"
raccourci = myNamespace.var1;
[...]
raccourci(); // ERREUR raccourci is not a
function
```

Mais tout ça n'est pas vraiment de notre ressort. Nous avons fait le maximum pour qu'il n'y ait pas de problème (on peut aussi mettre à disposition une documentation indiquant aux utilisateurs ce qu'il faut faire ou ne pas faire pour que ça marche au mieux). Au-delà de la conception de nos bibliothèques nous ne pouvons garantir aux utilisateurs le bon fonctionnement de leurs programmes s'ils n'utilisent pas correctement nos outils.

- Raccourcis d'utilisation (var toto = namespace.ns1.ns2 ;) équivalent de using namespace std ; en C++

3. Exemples

Et voilà, nous en sommes à présent à mettre en application tout ce que vous avez appris dans cet article. Je vous propose de mettre en oeuvre une mini bibliothèque illustrative et un cas d'utilisation, ce qui vous permettra d'appréhender dans sa globalité la notion d'espace de noms en JavaScript.

3.1. Votre bibliothèque : myOwnJSLibrary.js

myOwnJSLibrary.js

```
<!-- <![CDATA[
var myOwnJSLibrary = {
  author : "Nourdine FALOLA",
  addEventListener :
function(o,typeEvent,callback){
    if (o.addEventListener){
      o.addEventListener(typeEvent,callback,false
);
    }
    else if (o.attachEvent){
      o.attachEvent("on" + typeEvent,callback);
    }
  },

  displayException : function(){
    if (arguments.length < 1){
      // pas de paramètres passés à la fonction
      return;
    }

    var exc = arguments[0];
    if (!(exc instanceof Object) || exc.message
=== undefined){
      // le paramètre n'est pas une exception
      return;
    }

    // on affiche toutes les propriétés de
l'exception
    // (elles diffèrent suivant le navigateur)
    var msg = "";
    for (var prop in exc){
      msg += exc[prop] + "\n";
    }
    alert(msg);
  }

};
// ]]> -->
```

3.2. myOwnJSLibrary.Math.js

myOwnJSLibrary.Math.js

```
<!-- <![CDATA[
(function(){
  var f = function(){
    if (typeof myOwnJSLibrary == "undefined"){
      alert("the namespace myOwnJSLibrary was
deleted.");
      return false;
    }
    else if (typeof myOwnJSLibrary.author ==
"undefined" || myOwnJSLibrary.author != "Nourdine
FALOLA"){
      alert("the namespace myOwnJSLibrary was
altered.");
      return false;
    }

    return true;
  }

  if (window.addEventListener){ // non-IE browser
    window.addEventListener("load",f,false);
  }
  else if (window.attachEvent){ // IE browser
    window.attachEvent("onload",f);
  }

  if (!f())
    return;

  // extends myOwnJSLibrary
```

```
  myOwnJSLibrary.Math = {

    PI : Math.PI,

    add : function(){
      try{
        switch(arguments.length){
          case 0:
            return undefined;
          case 1:
            if (typeof arguments[0] != "number"){
              return undefined;
            }
            return arguments[0];
          default:
            var ret = 0;
            for (var i=0; i<arguments.length; i+
+){
              if (typeof arguments[i] !=
"number"){
                return undefined;
              }
              ret += arguments[i];
            }
            return ret;
        }
      }
      catch(exc){
        myOwnJSLibrary.displayException(exc);
      }
    }
  };

})(); // run the anonymous function

// ]]> -->
```

3.3. myOwnJSLibrary.Math.Statistic.js

myOwnJSLibrary.String.js

```
<!-- <![CDATA[
(function(){
  var f = function(){
    if (typeof myOwnJSLibrary == "undefined"){
      alert("the namespace myOwnJSLibrary was
deleted.");
      return false;
    }
    else if (typeof myOwnJSLibrary.author ==
"undefined" || myOwnJSLibrary.author != "Nourdine
FALOLA"){
      alert("the namespace myOwnJSLibrary was
altered.");
      return false;
    }

    return true;
  }

  if (window.addEventListener){ // non-IE browser
    window.addEventListener("load",f,false);
  }
  else if (window.attachEvent){ // IE browser
    window.attachEvent("onload",f);
  }

  if (!f())
    return;

  // extends myOwnJSLibrary

  myOwnJSLibrary.String = {

    add : function(){
      try{
        switch(arguments.length){
          case 0:
            return undefined;
          case 1:
            return "" + arguments[0];
          default:
            var ret = "";
            for (var i=0; i<arguments.length; i+
+){
              ret += arguments[i];
            }
            return ret;
        }
      }
      catch(exc){
        myOwnJSLibrary.displayException(exc);
      }
    },

    format : function(){
      switch(arguments.length){
        case 0:
          return undefined;
        case 1:
          return "" + arguments[0];
        default:
          var ret = arguments[0];
          var expr = "\\{n\\}";

          for (var i=1; i<arguments.length; i++){
            var motif = new
RegExp(expr.replace("n",i-1),"g");
            ret = ret.replace(motif,"" +
arguments[i]);
          }
          return ret;
      }
    }

  };

})(); // run the anonymous function

// ]]> -->
```

Retrouvez la suite de l'article de Nourdine Falola en ligne : Lien38

Développement d'un gestionnaire de contacts (CManager) avec ActionScript Foundry (AS Foundry) - Partie 1

Dans ce premier tutoriel, nous mettrons en évidence les concepts fondamentaux MVC et ceux de l'AS Foundry.

1. Introduction

Il existe aujourd'hui un nombre important d'initiatives open-source destinées au développement d'applications Flex. Le framework AS Foundry a été conçu en 2005 puis proposé à la communauté open source en 2007 par l'équipe de développement de ServeBox.

Afin d'illustrer son fonctionnement, nous allons créer pas-à-pas une application de gestion de contacts. CManager est constitué d'un client Flex et d'un service Java/BlazeDS exécuté sous Tomcat. Basé sur une structure simple, CManager permet la gestion (ajout, modification, suppression) de fiches contacts récupérées depuis le service Java.

Dans ce premier tutoriel, nous mettrons en évidence les concepts fondamentaux MVC et ceux de l'AS Foundry. D'autres tutoriaux permettront d'enrichir le CManager et illustreront certaines des fonctionnalités avancées de la ToolBox AS Foundry.

Vous avez un aperçu de CManager à l'adresse suivante : (Lien39)

La documentation de l'API est disponible ici (Lien40).

2. Prérequis

Environnement de développement :

- Eclipse 3.3.2 : (Lien41)
- Flex Builder Plugin 3.2 pour Eclipse : (Lien42)

Environnement d'execution :

- JDK 1.5
- Tomcat 6.x : nous avons préparé des versions préconfigurées de Tomcat pour Windows et Linux, disponibles ci-dessous.

3. Installation

3.1. Installation du JDK 1.5

Télécharger et installer le JDK 5,0 Update x (Lien43).

Créer la variable d'environnement correspondant à votre installation, par exemple :

```
${JAVA_HOME} = C:\Program Files\Java\jdk1.5.0_17
```

Rajouter la variable dans votre path :

```
Path= ... ;%JAVA_HOME%\bin
```

3.2. Installation de Tomcat

Télécharger apache-tomcat-6.0.16.zip (Lien44).

Dézipper l'archive dans un répertoire, par exemple : C:\Foundry\apache-tomcat-6.0.16

Créer la variable d'environnement :

```
${CATALINA_HOME} = C:\Foundry\apache-tomcat-
6.0.16
```

Rajouter la variable dans votre path :

```
Path= ... ;%CATALINA_HOME%\bin
```

Votre serveur d'application Apache Tomcat est prêt, lancez le grâce à startup.bat (C:\Foundry\apache-tomcat-6.0.16\bin).

3.3. Paramétrage de Flex Builder ou Eclipse

Téléchargez la structure de l'application CManager (Lien45) (format zip, 1 049 ko) et complétez le code source en suivant le tutoriel pas à pas.

Dans le fichier .actionScriptProperties de *flex-gui* remplacer l'attribut outputFolderLocation de la balise compiler :

```
outputFolderLocation="/C:/Foundry/apache-tomcat-
6.0.16/webapps/contactApplication/flex-gui-debug"
```

Toujours la même manipulation pour l'attribut serverRoot de la balise flexProperties dans le fichier .flexProperties :

```
serverRoot="C:\Foundry\apache-tomcat-
6.0.16\webapps\contactApplication"
```

Dans le fichier .project de *flex-gui* remplacer la balise location afin de pointer sur le dossier de déploiement dans Apache Tomcat :

```
<location>/C:/Foundry/apache-tomcat-
6.0.16/webapps/contactApplication/flex-gui-
debug</location>
```

Vous pouvez désormais importer le projet *flex-gui* dans votre workspace.

4. Le projet

L'application CManager est composée de trois sous-projets :

- « flex-gui » pour l'interface graphique Flex
- « java-service » pour le service Java
- « web-app » qui permet l'assemblage de l'application Web déployée sous Tomcat. Si vous utilisez la version de Tomcat fournie avec cet exemple, vous n'aurez pas besoin de vous intéresser à cette partie pour réaliser ce tutoriel.

4.1. Structure physique

Le projet « Java-service » contient les classes Java nécessaires à l'exécution du service pour la récupération des données. Il embarque une base de données HSQLDB qui s'installe automatiquement dans un répertoire temporaire du serveur Tomcat. Nous ne nous étendrons pas sur la partie service Java mais nous nous concentrerons sur le projet « flex-gui ».

Le dossier src/main/flex contient les sources.

Le dossier src/main/resources contient les ressources additionnelles utilisées par le compilateur : assets et feuilles de styles CSS.

4.2. Structure logique : Modèle, vue et contrôleur

Basée sur différents design patterns, l'AS Foundry permet un cycle de développement réduit grâce à l'utilisation d'outils tels que la synchronisation de données modèles-vues, navigation inter-écrans, liste de contrôle d'accès, internationalisation et externalisation de labels, et bien d'autres !

Dans le pattern MVC, le modèle prend en charge les données de l'application, les vues servent à l'affichage de ces données et au support des interactions de l'utilisateur. Enfin, le contrôleur traduit les actions de l'utilisateur en modifiant les données du modèle ou en déclenchant des appels aux services distants.

Pour plus d'information sur MVC, consultez l'article « The MVC Framework » (Lien46) de l'ASFoundry.

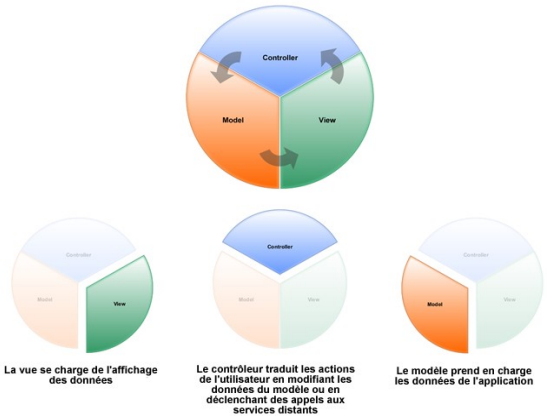
Les principaux packages AS Foundry sur lesquels nous allons intervenir sont les suivants :

- view : layout graphique
- helper : logique des vues
- control : contrôleur
- business : services distants
- model : modèle de données

5. Initialisation

5.1. Création du modèle

La partie modèle du framework AS Foundry est représentée par la classe AbtractModel. Cette classe est une implémentation de IObservable (sujet dans le design pattern Observer). Le rôle d'un Observable est de transmettre les modifications à une ou plusieurs instances d' « Observer » (dans notre cas les vues) en utilisant un mécanisme de notification.



Modèle du framework AS Foundry

Pour créer notre modèle, nous étendons la classe AbstractModel fournie par le framework.

```
//model/ContactModel.as
...
public class ContactModel extends AbstractModel
{
    public function ContactModel()
    {
        super();
    }
}
```

Pensez à bien vérifier que tous les imports de classe sont faits.

ContactModel va gérer la liste de contacts. Le getter et le setter permettent respectivement de récupérer et de modifier cette liste de contacts. L'utilisation de la méthode notifyObservers permet aux observateurs d'être informés des modifications intervenant sur la liste de contacts. Les observateurs (nos vues) pourront s'abonner au modèle et être notifiés des modifications intervenant sur nos données.

```
//model/ContactModel.as
...
private var _contacts : ArrayCollection;
...
public function getContacts(): ArrayCollection
{
        _contacts.source.sortOn("lastName");
        return _contacts;
}
public function setContacts( ar : ArrayCollection
): void
{
        _contacts = ar;
        // Notification des observateurs
éventuels
        notifyObservers( new
ContactListNotification( null ) );
}
...
```

Pensez à bien vérifier que tous les imports de classe sont faits.

5.2. Création du contrôleur

Le contrôleur permet de traiter l'ensemble des actions réalisées par l'utilisateur depuis les vues en faisant appel au modèle ou aux services distants. Pour créer le contrôleur de l'application, on étend simplement la classe AbstractController.

Ce contrôleur étant le point d'entrée principal de l'application :

- il doit implémenter le design pattern singleton afin que l'on puisse toujours obtenir la même instance ;
- il est chargé d'initialiser les modèles et les délégués des services distants (voir 7.3 Création du BusinessDelegate) : pour ce faire, nous surchargeons les méthodes initializeModels et initializeDelegates.

Le code suivant initialise ces méthodes pour créer l'instance de MainController. On peut noter que notre modèle est enregistré auprès du ModelLocator. ModelLocator implémente un Singleton, cette classe enregistre des références aux modèles de l'application.

```
//control/MainController.as
...
public class MainController extends
AbstractController
{
        private static var
_mainControllerInstance : MainController;
        ...
        /**
        * MainController Singleton
        * */
        public static function getInstance() :
MainController
        {
                if(_mainControllerInstance ==
null)
                {
                        _mainControllerInstance =
new MainController();
                }
                return _mainControllerInstance;
        }
        ...
        /**
        * Initialisation des Modèles
        * */
        override public function
initializeModels():void
        {
                ModelLocator.getInstance().regist
erModel( getQualifiedClassName(ContactModel), new
ContactModel() );
        }
        ...
        /**
        * Initialisation des services distants
        * */
        override public function
initializeDelegates():void
        {
                //Initialisation des services
distants
        }
        ...
```

Pensez à bien vérifier que tous les imports de classe sont faits.

Maintenant que votre Modèle et votre Contrôleur sont créés, nous pouvons initialiser l'application puis commencer à créer nos vues.

5.3. Initialisation de l'application

Dans le cas d'une application Flex basique, la racine est une instance de mx.core.Application. Dans le cas de l'AS Foundry, l'application doit effectuer l'initialisation de notre contrôleur. Nous étendons donc la classe AbstractApplication en redéfinissant la méthode *getControllerClass*.

```
//control/MainApplication.as
...
package
org.servebox.sample.contactapplication.control
{
    import
org.servebox.foundry.control.AbstractApplication;
    public class MainApplication extends
AbstractApplication
    {
        public function MainApplication()
        {
            super();
        }
        override protected function
getControllerClass():Class
        {
            return MainController;
        }
    }
}
```

Notre application est prête, il suffit maintenant de la déclarer comme composant racine, dans le fichier main.mxml.

```
<?xml version="1.0" encoding="utf-8"?>
<control:MainApplication
        xmlns:mx="http://www.adobe.com/2006/mxml"
        xmlns:control="org.servebox.sample.contac
tapplication.control.*"
        xmlns:view="org.servebox.sample.contactap
plication.view.*"
        >
        <mx:Style
source="/../resources/css/main.css"/>
</control:MainApplication>
```

6. Créer une vue

Pour créer une vue, on ajoute un fichier MXML en utilisant comme balise racine, un des composants disponibles de l'AS Foundry (CanvasView, HBoxView, VBoxView).

L'application est constituée d'une vue avec plusieurs « State » (MainView.mxml) :

- BASE_VIEW_STATE permettant la consultation et la modification d'un contact.
- ADD_CONTACT_VIEW_STATE permettant l'ajout d'un contact.

Nous n'allons pas nous attarder sur le code source de la vue, qui ne contiendra que des composants graphiques (le code source de la vue est disponible dans le package view). Nous allons plutôt nous intéresser au ViewHelper qui va permettre à notre vue d'interagir avec le reste de l'application.

6.1. Le ViewHelper

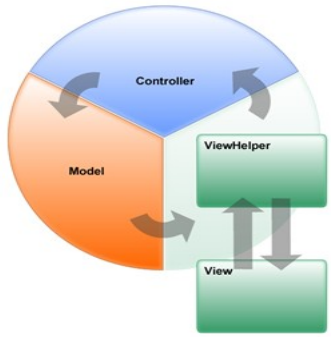
La seule utilisation de MXML pour décrire les vues n'est pas conseillée, même si cela peut paraître plus facile au premier abord. Le code source devient vite difficile à maintenir car le de layout et le code de gestion sont mélangés.

Le framework AS Foundry propose une séparation entre la présentation et la logique fonctionnelle. Chaque vue est associée à un ViewHelper, contenant:

- la logique fonctionnelle
- les appels au contrôleur
- les événements
- la gestion des notifications provenant du modèle et les « data binding »

Ainsi la vue peut facilement être décrite en utilisant les balises MXML. La logique fonctionnelle étant séparée, le code sera plus facile à maintenir, réutiliser ou encore étendre.

Nous créons donc la classe MainViewHelper qui contient un « binding » vers la liste des contacts. En tant qu'implémentation de l'interface IObserver, notre ViewHelper peut s'enregistrer auprès de ContactModel et être notifié en surchargeant la méthode *update*.




```
//helper/MainViewHelper.as
package
org.servebox.sample.contactapplication.helper
{
import
org.servebox.sample.contactapplication.model.Cont
actModel;
import flash.events.Event;
import flash.utils.getQualifiedClassName;
import org.servebox.foundry.model.ModelLocator;
import
org.servebox.foundry.observation.IObservable;
import
org.servebox.foundry.observation.Notification;
import org.servebox.foundry.view.ViewHelper;
import
org.servebox.sample.contactapplication.control.Ma
inController;
import
org.servebox.sample.contactapplication.model.noti
fication.ContactListNotification;
import
org.servebox.sample.contactapplication.service.va
lue.vo.ContactVO;
import
org.servebox.sample.contactapplication.view.MainV
iew;
public class MainViewHelper extends ViewHelper
{
...
        /**
        * S'enregistre comme un observateur
auprés de ContactModel
        * et permet de recevoir les notifications
depuis ContactModel
        * */
        override public function
registerToModels():void
        {
                var model : IObservable =
ModelLocator.getInstance().getModel(getQualifiedC
lassName( ContactModel ));
                model.registerObserver(this);
        }
        /**
        * Déclenché aprés une notification de
ContactModel
        * */
        override public function
update( o:IObservable, n:Notification) :void
        {
                // réalise une action aprés avoir
reçu une notification depuis ContactModel
                dispatchEvent( new
Event("contactListChange") );
        }
}
...
```

Pensez à bien vérifier que tous les imports de classe sont faits.

6.2. Ajout de la vue à l'application

Avant d'ajouter la vue à l'application, il faut lui associer son « helper ». Il suffit d'ajouter simplement cette balise dans le code MXML:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- view/MainView.mxml -->
<view:HBoxView
        xmlns:view="org.servebox.foundry.view.*"
        xmlns:mx="http://www.adobe.com/2006/mxml"
        width="100%"
        height="100%"
        horizontalAlign="center"
        horizontalCenter="0">
        <!-- association de la vue avec
MainViewHelper.as -->
        <helper:MainViewHelper id="helper"
autoRegisterBasedOnQualifiedClassName="true"/>
...
</view:HBoxView>
```

On ajoute la vue à notre application, en la plaçant dans un « ViewStack » par exemple :

```
<?xml version="1.0" encoding="utf-8"?>
<control:MainApplication
        xmlns:mx="http://www.adobe.com/2006/mxml"
        xmlns:control="org.servebox.sample.contac
tapplication.control.*"
```

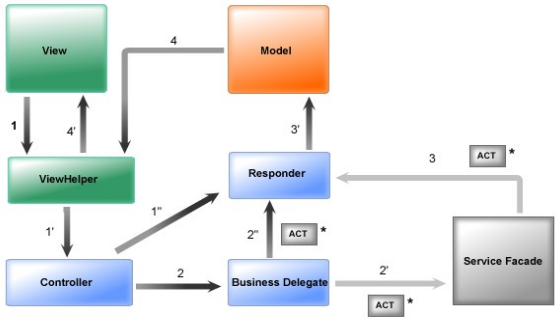
```
        xmlns:view="org.servebox.sample.contactapplication.view.*"
        >
        <mx:Style source="/../resources/css/main.css"/>
        <mx:ViewStack id="myViewStack" width="1024" height="720">
                <view:MainView id="mainView"/>
        </mx:ViewStack>
</control:MainApplication>
```

7. Connecter une vue au service

Pour charger nos contacts, l'application va déclencher la séquence d'opérations suivantes :

- 1 - Au chargement de l'application, le ViewHelper est appelé pour récupérer la liste des contacts.
- 1' - Il appelle la méthode getContactList dans le contrôleur.
- 1" - Le contrôleur crée une instance du Responder (expliqué ci-dessous).
- 2 - Le contrôleur demande au BusinessDelegate de commencer l'appel au service distant.
- 2' - Le BusinessDelegate appelle le service et récupère un ACT (Asynchronous Completion Token).
- 2" - Le BusinessDelegate renvoie l'ACT à l'instance du Responder.
- 3 - Le service répond et le Responder associé à l'ACT est appelé.
- 3' - Le Responder gère la réponse du service en modifiant la valeur dans le modèle.
- 4 - Une notification est envoyée aux observers
- 4' - Le ViewHelper peut alors récupérer les données et déclencher la mise à jour de la vue.

Asynchronous Completion Token (ACT): les outils de connectivité Flex utilisent ce design pattern, on associe les actions et les états de l'application avec des réponses qui indiquent que les opérations asynchrones sont terminées. Pour chaque opération asynchrone, on crée un ACT qui identifie les actions et les états requis pour le résultat de l'opération. Quand le résultat est renvoyé, on peut utiliser cet ACT pour le différencier des résultats des autres opérations asynchrones. Le client utilise l'ACT pour identifier l'état requis pour gérer le résultat.



7.1. Traiter la réponse du service (responder)

Le rôle principal d'un Responder est de gérer les réponses d'une méthode spécifique provenant du service. Chaque Responder doit implémenter l'interface IbusinessResponder. Les méthodes *fault* et *result* sont destinées respectivement à gérer les erreurs systèmes (erreur réseau par exemple) ou propager les résultats. Dans le cas présent, la méthode *résult* de *ContactListResponder* fait appel au setter de *ContactModel*.

```
//business/responder/ContactListResponder.as
...
public class ContactListResponder implements IBusinessResponder
{
        public function ContactListResponder()
        {}
        public function result(data:Object):void
        {
                // Ici on gère la réponse du service distant
        }
        public function fault(info:Object):void
        {
                Alert.show("Error getting contact list !");
        }
}
```

7.2. Création du BusinessDelegate

Avant d'ajouter la fonction au contrôleur, nous devons créer notre « businessDelegate ». Lorsque vous utilisez l'AS Foundry, il faut créer une instance de BusinessDelegate, afin de permettre les interactions entre le service distant et le design pattern MVC. L'un de principaux avantages réside dans le fait que les changements sur le service distant ne poseront de conflits que sur le BusinessDelegate, et aucun autre composants de l'application. Il y a d'autres avantages à utiliser les « businessDelegate », pour avoir plus d'informations lisez ce document (Lien47).
Il faut donc créer *MainBusinessDelegate* dans le package com.servebox.sample.contactapplication.business. Puis on crée la méthode getContactList, à l'intérieur de cette méthode on relie l'ACT (asynchronous completion token) à une instance de IbusinessResponder. Le « responder » gère la réponse du service.

```
//business/MainBusinessDelegate.as
...
public class MainBusinessDelegate extends AbstractBusinessDelegate implements IBusinessDelegate
{
        public function MainBusinessDelegate( service : Object=null )
        {
                super(service);
        }
        /**
        * Appel la requete getContactList() du service Java
        * */
        public function getContactList( responder : ContactListResponder ) : void
        {
                linkResponderToCallToken( getService().getContactList(), responder );
        }
...
```

On initialise le « businessDelegate » en surchargeant la méthode initializeDelegates dans le contrôleur. L'appel au serveur distant est réalisé par le contrôleur en utilisant la classe ContactListResponder (que nous allons créer par la

suite) en tant que « responder ».

```
//control/MainController.as
...
/**
* Récupère le service Java
* */
private function get service() : RemoteObject
{
        var service : RemoteObject = new
RemoteObject("java-service");
        service.channelSet =
ChannelSetProvider.getInstance().getDefaultChanne
lSet();
        return service;
}
/**
* Initialisation du service distant
* */
override public function
initializeDelegates():void
{
        var bd : MainBusinessDelegate = new
MainBusinessDelegate( service );
        registerBusinessDelegate( bd,
getQualifiedClassName( MainBusinessDelegate ) );
}
/**
* Appel à la fonction getContactList du
MainBusinessDelegate
* */
public function getContactList() : void
{
        getMainBusinessDelegate().getContactList(
new ContactListResponder() );
}
...
```

Pensez à bien vérifier que tous les imports de classe sont faits.

7.3. Afficher les données

Nous commençons par créer une méthode dans MainViewHelper pour appeler la méthode *getContactList* dans le contrôleur.

```
//helper/MainViewHelper.as
...
        /**
        * Appel la méthide getContactList dans
MainController
        * */
        public function retrieveContactList() :
void
        {
                MainController.getInstance().getC
ontactList();
        }
...
```

On crée le « binding » sur le getter dans MainViewHelper, pour alimenter la liste. Si vous n'êtes pas familier avec le binding, lisez ce document: (Lien48).

```
//helper/MainViewHelper.as
...
        /**
        * Renvoi un tableau des contacts stockés
dans le model
        * */
        [Bindable("contactListChange")]
        public function get getContactList() :
Array
        {
                return
getContactModel().getContacts().source;
        }
        /**
        * Accesseur de ContactModel
        * */
        public function getContactModel() :
ContactModel
        {
                return
ContactModel(ModelLocator.getInstance().getModel(
getQualifiedClassName( ContactModel )));
        }
...
```

Pensez à bien vérifier que tous les imports de classe sont faits.
Nous mettons à jour *MainView* et son *MainViewHelper*. Il suffit ensuite d'ajouter la propriété DataProvider au composant List et un bouton pour récupérer la liste des contacts.

```
//view/MainView.mxml
...
        <HBoxView
                xmlns="org.servebox.foundry.view.
*"
                xmlns:mx="http://www.adobe.com/20
06/mxml"
                xmlns:comp="org.servebox.sample.c
ontactapplication.component.*"
                width="{MainConf.SCENE_WIDTH}"
height="{MainConf.SCENE_HEIGHT}"
                xmlns:form="org.servebox.toolbox.
form.*"
                xmlns:elements="org.servebox.tool
box.form.elements.*"
                xmlns:helper="org.servebox.sample
.contactapplication.helper.*"
                horizontalAlign="center"
                horizontalCenter="0"
                horizontalScrollPolicy="off"
                verticalScrollPolicy="off"
        >
...
        <mx:Button
                id="getBtn"
                icon="{EmbeddedMedia.getInstance(
).updateContactImg}"
                click="{helper.retrieveContactLis
t()}"
        />
...
        <mx:List
                width="300"
                height="648"
                id="contactList"
                dataProvider="{helper.getContactL
ist}"
        />
...
```

Retrouvez la suite de l'article de Wajdi Hadj ameur en ligne : Lien49

Boost 1.38 et MingW : un mariage qui prend du temps !

Différents intervenants du site ont rencontré un plantage lors de l'utilisation de certaines bibliothèques Boost. Le plantage se traduit par l'exception *"Exception: could not convert calendar time to [XXX] time"*. Comme toujours, les warning du compilateur mettent la puce à l'oreille. En effet, le simple code suivant permet de reproduire le plantage :

```
#include <iostream>
#include
"boost/date_time/posix_time/posix_time.hpp"

int main()
{
   try{
      boost::date_time::microsec_clock<boost::pos
ix_time::ptime>::local_time();
   }
   catch(std::exception& e) {
      std::cout << "  Exception: " <<  e.what()
<< std::endl;
   }
    return 0;
}
```

Or, le compilateur nous indique :

```
/boost_1_38_0/boost/date_time/
filetime_functions.hpp: In function `uint64_t
boost::date_time::winapi::file_time_to_microsecon
ds(const FileTimeT&)':
/boost_1_38_0/boost/date_time/
filetime_functions.hpp:101: warning: left shift
count >= width of type
```

Intéressons nous à ces lignes de codes suspectes (fichier boost_1_38_0/boost/date_time/filetime_functions.hpp) :

```
    template< typename FileTimeT >
    inline boost::uint64_t
file_time_to_microseconds(FileTimeT const& ft)
    {
        /* shift is difference between 1970-Jan-
01 & 1601-Jan-01
        * in 100-nanosecond intervals */

        const uint64_t c1 = 27111902UL;
        const uint64_t c2 = 3577643008UL; //
issues warning without 'UL'
        const uint64_t shift = (c1 << 32) + c2;

        union {
            FileTimeT as_file_time;
            uint64_t as_integer; // 100-nanos
since 1601-Jan-01
        } caster;
        caster.as_file_time = ft;

        caster.as_integer -= shift; // filetime
is now 100-nanos since 1970-Jan-01
        return (caster.as_integer / 10); //
truncate to microseconds
    }
```

Un bon debugger avec un point d'arrêt placé pertinemment mettent en évidence que le calcul est faux : *shift* prend la valeur de *c2* ! Il semblerait que le compilateur optimise (même en mode debug) le calcul un peu trop rapidement. Les valeurs constantes *c1* et *c2* sont prises en 32 bits pour le calcul, en particulier pour le décalage de 32 bits de *c1*. Ce qui explique la valeur de *shift* égale à celle de *c2*. Le résultat est donc erroné provoquant l'exception ultérieurement.
Nous proposons de modifier les lignes comme suit (UL est remplacé par ULL) :

```
    template< typename FileTimeT >
    inline boost::uint64_t
file_time_to_microseconds(FileTimeT const& ft)
    {
        /* shift is difference between 1970-Jan-
01 & 1601-Jan-01
        * in 100-nanosecond intervals */
        const uint64_t c1 = 27111902ULL;
        const uint64_t c2 = 3577643008ULL; //
issues warning without 'UL'
        const uint64_t shift = (c1 << 32) + c2;

        union {
            FileTimeT as_file_time;
            uint64_t as_integer; // 100-nanos
since 1601-Jan-01
        } caster;
        caster.as_file_time = ft;

        caster.as_integer -= shift; // filetime
is now 100-nanos since 1970-Jan-01
        return (caster.as_integer / 10); //
truncate to microseconds
    }
```

Remarquons le commentaire des auteurs de boost :

```
// issues warning without 'UL'
```

Il semblerait que les auteurs ont rencontré un problème similaire mais que la correction n'a pas été complète.

Le plantage pourrait concerner les bibliothèques suivantes :

- **Boost.Asio**
- **Boost.Thread**
- **Boost.Date_Time**
- **Boost.Interprocess**
- **Boost.Statechart**

A ce jour, le plantage n'a été détecté que sur les deux premières bibliothèques avec le compilateur mingw 3.4.2.
Attention à recompiler les bibliothèques une fois la modification apportée dans le fichier.

Tutoriel Boost.Asio

Cet article introduit la programmation réseau en C++ à l'aide de Boost.Asio. Après un rapide tour d'horizon de l'architecture globale de Boost.Asio et des possibilités offertes par cette bibliothèque (opérations synchrones et asynchrones), cet article présentera les Timers, la communication TCP et UDP. Des exemples concrets de clients et serveurs seront étudiés. Enfin, un projet réseau réaliste avec un code robuste sera présenté en dernière partie.

1. Introduction

1.1. Réseau et langage de programmation

En langage C++ (et C), la programmation réseau est dépendante du type des machines et systèmes d'exploitation utilisés. Ce qui ne rend pas facile la tâche du programmeur... Par exemple, il est souvent fastidieux de réaliser une version Windows et Linux, car s'il existe de nombreux points communs avec POSIX (socket, bind, connect, listen, accept), il existe aussi de nombreuses divergences dans les en-têtes et les fonctions d'initialisations. Si bien que développer une application réseau portable peut rapidement devenir un véritable challenge pour le programmeur.

Boost.Asio répond à ces problèmes en proposant une bibliothèque de haut niveau portable, facile à utiliser et dans un style C++ très élégant.

1.2. Pré-requis

Les concepts C++ présentés dans une première partie sont relativement simples. La dernière partie fait toutefois appel à des notions intermédiaires nécessaires au développement d'applications robustes, comme les pointeurs intelligents (Lien50) [Loïc Joly] ou encore la sérialisation (Lien51) [Pierre Schwartz]. Le lecteur est vivement encouragé à lire les articles correspondants en cas de difficultés.

La compréhension globale du fonctionnement d'un réseau n'est pas obligatoire, mais conseillée. Le lecteur trouvera parmi les liens suivant de bonnes références :

- les sockets en C (Lien52) [Benjamin Roux],
- la théorie des réseaux locaux et étendus (Lien53) [Patrick Hautrive],
- Initiation à la programmation réseau sous Windows (Lien54) [Jessee Edouard]

1.3. Installation de Boost.Asio

Boost.Asio fait parti de la grande bibliothèque Boost. Les exemples de cet article ont été compilés avec VC++ Express 2008 et Boost 1.37. Pour pouvoir utiliser Boost.Asio, il est conseillé d'installer boost.regex, boost.thread, boost.date_time et boost.serialization. Voici quelques liens expliquant comment installer boost, soit à l'aide d'un exécutable (Windows et Visual uniquement), ou bien en compilant à l'aide de bjam:

- Installer et utiliser Boost/Boost.TR1 avec Visual C++ (Lien55) [Aurélien Regat-Barrel]
- Compilation de Boost (Lien56) [ram-0000]

Pour ne prendre que l'utile pour ce tutoriel, compilez boost avec la ligne suivante:

```
bjam --with-system --with-thread --with-date_time
--with-regex --with-serialization stage
```

1.4. Code et commentaires dans ce tutoriel

Dans la mesure du possible, le code ne sera pas coupé par des commentaires, afin de ne pas nuire à la lisibilité du programme dans son ensemble. Ainsi, j'ai choisi dans la plupart des cas de décrire le programme d'abord, avec des références sur l'endroit du code entre parenthèses. Cela n'empêche évidemment pas de laisser quelques commentaires dans le code, comme tout bon programme.
Pour plus de lisibilité, les noms de fonctions/classes seront en italiques dans tout l'article (excepté dans le code) ainsi que les termes anglais.

Exemple:

La fonction *ma_fonction()* prend deux chaines de caractères en paramètres (*std::string*).

On commence par afficher les deux chaines (1).

On stocke la somme des deux chaines dans une variable temporaire (2).

On lance la fonction *cherche_char* sur chaque caractère (3).

```
void ma_fonction(const std::string& str1, const
std::string& str2)
{
    // On affiche les chaines // (1)
    std::cout << str1 << std::endl;
    std::cout << str2 << std::endl;

    // On stocke le tout dans une autre chaine //
(2)
    std::string chaine = str1 + str2;

    // On recherche des caractères précis // (3)
    std::for_each(chaine.begin(), chaine.end(),
cherche_char);
}
```

2. Les bases de Boost.Asio

Dans cette partie, nous allons découvrir ensemble les bases de Boost.Asio, à savoir les classes principales, les opérations synchrones et les opérations asynchrones.

Quelque soit la partie de Boost.Asio que l'on utilise, le

programme devra toujours posséder sa pièce centrale : un **io_service**. C'est en quelque sorte le "cœur" de la bibliothèque...

```
#include <boost/asio.hpp>

int main()
{
    boost::asio::io_service io_service;
    //Code...
}
```

Nous pouvons également spécifié explicitement à Boost Asio la plateforme cible. Il se peut qu'elle soit différente de la plateforme de développement.

Boost.Asio et Windows XP
```
#define _WIN32_WINNT 0x0501
```

Boost.Asio et Windows 2000
```
#define _WIN32_WINNT 0x0500
```

Pour rappel, il existe deux grands types d'opérations réseaux : les opérations synchrones et les opérations asynchrones. Nous allons détailler ces deux types d'opérations dans cette partie, avec leurs avantages et inconvénients.

2.1. Opérations synchrones

Une opération synchrone bloque la fonction appelante jusqu'à ce qu'elle aie terminé. Considérons le morceau de code suivant, qui envoie un *msg* par l'intermédiaire d'une *socket*:

Envoi synchrone
```
void Mafonction()
{
    //Code...
    boost::asio::send(socket,
boost::asio::buffer(msg)); // Envoi des données
par le socket
    std::cout << "Terminé" << std::endl;
}
```

Le message "Terminé" ne sera affiché à l'écran que lorsque la fonction d'envoi de donnée sera terminée. On dit que la fonction est **bloquante**. Dans cet exemple, la fonction *send()* pourrait ne pas bloquer longtemps car il n'y a pas besoin d'attendre pour envoyer des données. Cependant, on pourrait imaginer des cas où on fait des demandes de send plus rapidement qu'elles ne sont transmises...
Autre exemple, que va t-il se passer si on effectue non pas un envoi, mais une réception de données?

Réception synchrone
```
void Mafonction()
{
    //Code...
    boost::asio::read(socket,
boost::asio::buffer(msg)); // Réception des
données sur le socket
    std::cout << "Terminé" << std::endl;
}
```

Dans ce code, la fonction *read()* va attendre la réception de donnée sur un port précis et on peut parfois attendre longtemps... Tant que la fonction *read()* n'a pas reçu une certaine quantité de donnée, la fonction appelante *Mafonction()* va rester bloquée.

Pour pallier à ce mode de fonctionnement, on pourrait lancer la réception de données dans un thread par exemple. On s'affranchirait ainsi du problème de la réception synchrone. Par contre, le développeur devra dans ce cas gérer les accès concurrents lui- même. Il existe une solution beaucoup sûre et plus élégante pour résoudre ce problème : les opérations asynchrones.

2.2. Opérations asynchrones

Des opérations asynchrones sont des opérations qui rendent la main immédiatement à l'appelant même si elles ne sont pas encore terminées. On sait quand elles commencent, mais on ne sait pas quand elles finissent, dans l'absolu. Par contre, elles appellent une fonction *callback* lorsqu'elles ont terminé. En effet, une opération asynchrone prend en général en paramètre un *completion_handler* (cf. exemple suivant) qu'elle va appeler lorsqu'elle a terminé son travail. Elle n'empêche pas la fonction appelante de continuer d'exécuter son code, on dit donc que l'opération est **non bloquante**.

Les *callback* sont gérés dans une **boucle de traitement** des événements. Cette boucle de traitement est instanciée par l'utilisateur à l'aide de la fonction **io_service::run()** . Cette fonction est bloquante tant qu'il reste des opérations asynchrones en cours et rend la main lorsqu'il n'y a plus de *callback* à exécuter. Il faut donc impérativement donner du travail asynchrone à réaliser avant d'appeler *io_service::run()*. Pour maintenir une IHM active, *io_service::run* peut être exécuté dans un thread dédié.

Sur certaines plateformes, l'implémentation de Boost.Asio peut éventuellement utiliser des threads **en interne** pour émuler l'asynchronicité. Ces threads restent invisibles pour l'utilisateur et la bibliothèque s'utilise comme si toutes les opérations étaient lancées dans un thread unique. Tant que le programme ne possède qu'un seul *io_service*, la boucle de traitement des événements est traité de manière **strictement séquentielle**. Le développeur n'a donc **pas** à gérer les accès concurrents.

Le prototype de la fonction *callback* de Boost.Asio varie suivant les opérations asynchrones. La fonction *async_read* prend deux paramètres, alors que la fonction *async_connect* n'en prend qu'un. Afin de mieux contrôler soi même les arguments passés au *callback*, on utilisera *boost::bind*.

Réception asynchrone
```
void completion_handler(const
boost::system::error_code& error)
{
    std::cout << "Terminé" << std::endl;
}
void Mafonction()
{
    //Lecture asynchrone
    boost::asio::async_read(socket,
boost::asio::buffer(msg),
        boost::bind(&completion_handler,
boost::asio::placeholders::error)
```

```
        );
    std::cout << "Ca s'affiche ! " << std::endl;
}
```

Attention, il est de la responsabilité du développeur de s'assurer que la durée de vie des structures/variables utilisées lors d'opérations asynchrones sont suffisantes. Même si Boost.Asio effectue des copies de buffer, la responsabilité de la mémoire sous jacente revient à l'appelant. La mémoire doit vivre jusqu'à l'appel du callback!
Le non respect de cette règle peut entrainer violation de mémoire et/ou comportement indéterminé parfois difficiles à déboguer.

En particulier, le buffer de réception ne doit pas être un buffer temporaire alloué sur la pile s'il est libéré avant la fin de l'exécution et le buffer d'envoi ne doit pas être libéré prématurément.

async_read() ne bloque pas la fonction appelante *Mafonction()*. Une fois que l'opération *async_read()* est terminée, notre fonction *completion_handler()* sera appelée pour nous signaler qu'*async_read()* a bien effectué son travail. Pour nous informer du bon déroulement ou non de l'opération asynchrone, il nous suffit d'inspecter le paramètre *error_code* de notre *callback*. Avec ce paramètre, on peut traiter de nombreux cas comme :

- Connexion refusée
- Connexion perdue
- Argument invalide
- Message trop long
- etc..

Réception asynchrone avec retour de code d'erreur

```
void completion_handler(const
boost::system::error_code& e)
{
    if (!error)
    {
        std::cout << "Tout s'est bien passé" <<
std::endl;
    }
    else {
        // Problème que l'on peut identifier
        // access_denied, connection_aborted, ...
    }
}
```

Pour pouvoir effectuer des opérations asynchrones, nous devons lancer la boucle de gestion des événements par la fonction *io_service::run()*:

```
boost::asio::io_service ios;
// .... opérations asynchrones
ios.run();
```

Rappel : la fonction *run()* est bloquante continue de "travailler" tant qu'il reste des opérations asynchrones en cours.

2.3. Synchrone / Asynchrone ?

Inconvénients des opérations asynchrones:

- Complexité du programme
- Consommation mémoire (car les buffers de réception doivent tous être alloués et indépendants)

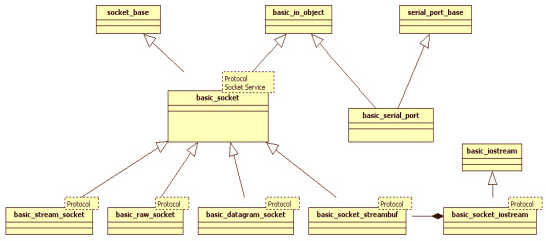
Avantages des opérations asynchrones:

- Performance
- Opérations non bloquantes

Le mode asynchrone sera beaucoup utilisé par la suite, en raison de ses performances et de ses appels non bloquants.

2.4. Architecture de Boost.Asio

Boost.Asio permet de gérer deux types de périphériques de communications : port Ethernet et port série. En ce qui concerne les réseaux, Boost.Asio permet l'utilisation de plusieurs protocoles : TCP (mode connecté), UDP (mode non connecté), ICMP. Le diagramme ci dessous permet d'apprécier la diversité des modes de communication qu'offre Boost.Asio.



Diagrammes des classes de Boost.Asio

Les classes d'utilisation communes sont fournies par Boost à l'aide d'un *typedef*, dans le bon *namespace*:

```
namespace tcp {
    //...
    typedef basic_stream_socket< tcp > socket;
    typedef basic_socket_iostream< tcp >
iostream;
}

namespace udp
{
    //...
    typedef basic_datagram_socket< udp > socket;
}
```

3. Les Timers

Boost.Asio propose un seul timer (le deadline_timer) qui fait tout ce qu'on lui demande, c'est à dire compter le temps, que ce soit de manière synchrone ou asynchrone. Le *deadline_timer* prend en paramètre un *io_service*, toujours indispensable, ainsi qu'une durée de référence.

Construction du timer Boost.Asio

```
// Construction d'un timer d'une seconde.
boost::asio::deadline_timer t(io,
boost::posix_time::seconds(1));
```

Le timer se déclenche au moment de sa construction et non pas à partir de l'attente !

3.1. Timers synchrones

Nous allons créer dans cette section un simple timer bloquant qui attend 5 secondes, puis rend la main au programme.

Timer synchrone

```
#include <iostream>
#include <boost/asio.hpp>
#include
<boost/date_time/posix_time/posix_time.hpp>

int main()
{
        boost::asio::io_service io; // Service
principal

        boost::asio::deadline_timer t(io,
boost::posix_time::seconds(5)); // Commence à
compter dès sa création
        t.wait(); // On attend que le timer
expire

        std::cout << "Terminé !" << std::endl;

        return 0;
}
```

La programme affiche donc "Terminé !" après avoir attendu les 5 secondes du timer.

3.2. Timers asynchrones

On souhaite maintenant que notre timer de la section précédente soit non bloquant. Pour cela, on va utiliser le mode asynchrone, déjà vu à la section précédente.

Le fonctionnement est très semblable à ce que nous avons déjà vu :

1. Création d'un timer avec un durée de compte à rebours de 5 secondes
2. Lancement en mode asynchrone du timer
3. Lorsque le timer est expiré, il appelle son *callback* : la fonction *print()*
4. La fonction *print()* affiche "Terminé !"

Timer asynchrone

```
#include <iostream>
#include <boost/asio.hpp>
#include
<boost/date_time/posix_time/posix_time.hpp>

void print(const boost::system::error_code&
/*error*/) // (3)
{
    std::cout << "Terminé !" << std::endl; // (4)
}

int main()
{
    boost::asio::io_service io;

    boost::asio::deadline_timer t(io,
boost::posix_time::seconds(5)); // (1)
    t.async_wait(print); // Attente asynchrone
(2)

    io.run();

    return 0;
}
```

Rappel : Il est important de toujours donner du travail asynchrone à effectuer AVANT d'appeler *io_service::run()*, sinon *io_service::run()* rendra la main immédiatement

Il peut arriver que l'on crée un timer pour s'en servir plus tard. Dans ce cas, il a de forte chance d'être dans l'état "expiré". La fonction membre *expires_at()* permet de changer la date d'expiration du timer, comme ceci:

```
boost::asio::deadline_timer t(io,
boost::posix_time::seconds(5));
// Code... le temps passe... le timer est déjà
expiré.

// On redonne au timer une nouvelle deadline:
t.expires_at(t.expires_at() +
boost::posix_time::seconds(5));

// Et notre timer est de nouveau en train de
compte ses 5 secondes.
t.async_wait(print); // Attente asynchrone
```

Nous verrons dans la prochaine section comment un timer peut être utilisé pour arrêter des opérations asynchrones en cours.

4. Le protocole TCP

Dans cette section, nous allons étudier en détail comment communiquer en TCP/IP avec Boost.Asio, en mode synchrone et asynchrone. Nous étudierons deux exemples d'architecture client/serveur. Un exemple beaucoup plus poussé sera présenté dans la dernière partie de ce tutoriel.

4.1. Introduction

Le protocole TCP s'utilise uniquement en mode connecté. Les fonctions et options classiques bas niveau POSIX sont disponibles, (*bind()*, *listen()*, SO_REUSEADDR, etc.) pour permettre au développeur de contrôler plus finement son application. Pour des applications "classiques", l'API haut niveau de Boost.Asio avec les valeurs par défaut suffisent amplement. C'est ce que nous allons utiliser ici.

Listons dans le tableau suivant les classes TCP de Boost.Asio les plus utilisées.

Les classes Boost.Asio	Ce qu'elles représentent
boost::asio::ip::tcp::endpoint	Représente un couple {adresse IP, port}
boost::asio::ip::tcp::resolver	Le résolver permet de construire un TCP endpoint a partir du nom d'un serveur
boost::asio::ip::tcp::socket	une socket, interface de communication avec le monde extérieur. Elle possède les fonctions membres *connect()* pour se connecter à un serveur. *send()* et *async_send()* pour envoyer des données par cette socket et *receive()* et *async_receive()* pour en recevoir.
boost::asio::ip::tcp::acceptor	C'est elle qui possède la fonction membre *accept()* et *async_accept()* pour accepter les connexions entrantes sur un serveur.

Un *endpoint* prend une adresse et un port dans le constructeur. Il existe plusieurs moyens pour le remplir correctement:

- Si on connait l'adresse IP du serveur, on va lui passer par une chaine de caractère.

Construction d'un endpoint à partir d'une adresse

```
tcp::endpoint
endpoint(boost::asio::ip::address::from_string("1
92.168.0.4"), 13);
```

- Si on ne connait que le nom DNS, il va falloir passer par un *resolver*. Dans l'exemple de code ci-dessous, on va récupérer une liste d'adresses IP correspondant à l'acronyme recherché.

On commence par créer un *resolver* (1) que l'on va utiliser pour récupérer **toutes les IP** correspondant au nom DNS fourni en entrée (ici www.developpez.com (Lien57)). Dans notre exemple, on va se connecter sur le port 80, qui est le port standard HTTP (2).

On lance la résolution (3). La fonction *resolver::resolve()* retourne un itérateur sur le premier *endpoint*. Libre à nous d'en faire ce que l'on veut, comme l'afficher par exemple (4).

Construction d'un ou plusieurs endpoint à partir d'un acronyme

```
#define _WIN32_WINNT 0x0501 // Asio et Windows XP
#include <boost/asio.hpp>
#include <iostream>

int main()
{
    // Création du service principal et du
résolveur.
    boost::asio::io_service ios;
    boost::asio::ip::tcp::resolver resolver(ios);
// (1)

    // Paramètrage du resolver sur Developpez.com
    boost::asio::ip::tcp::resolver::query
query("www.developpez.com", "80"); // (2)

    // On récupère une "liste" d'itérateur
    boost::asio::ip::tcp::resolver::iterator iter
= resolver.resolve(query); // (3)
    boost::asio::ip::tcp::resolver::iterator end;
//Marqueur de fin
    while (iter != end) // On itère le long des
endpoints
    {
        boost::asio::ip::tcp::endpoint endpoint =
*iter++;
        std::cout << endpoint << std::endl; // on
affiche (4)
    }

    return 0;
}
```

www.developpez.com possède une seul IP, contre 3 pour www.google.fr, ce qu'on peut observer dans le tableau suivant :

Acronyme	Sortie écran
www.developpez.com	87.98.130.52:80
www.google.fr	66.102.9.104:80 66.102.9.147:80 66.102.9.99:80

Les *endpoint* sont très utiles et importants puisqu'ils permettent de travailler indépendamment du type d'adresses utilisées : IPv4 et IPv6.

4.2. Lecture / écriture courte et transfert complet

Que ce soit en mode synchrone ou asynchrone, il existe deux types de communication *via* les sockets:

- Les transferts courts : Les messages transmis ne comportement pas de délimitations. Ces appels réseaux peuvent transmettre moins d'informations que le contenu original. Ce phénomène peut être dû au contrôle du flux du protocole de transport, ou bien à la bufferisation des données. Au final, on récupérera bien l'intégralité des données, mais en plus ou moins de morceaux qu'au départ. A titre d'exemple, 2 trames envoyées peuvent se traduire aussi bien par 1 ou 2 ou 3 réceptions.
- Les transferts complets: Dans ces appels réseaux, il faut préciser exactement la taille des données que l'on souhaite envoyer et recevoir. Dans le cas de la réception particulièrement, les données ne sont rendues disponibles (appel du *callback*) que lorsque le buffer de réception est **plein**. Ce mode de fonctionnement est particulièrement indiqué pour des messages de taille fixe ou lorsqu'on indique d'abord par un premier message de taille fixe la taille des données que l'on doit recevoir. Beaucoup d'applications réseaux ont besoin de ces garanties pour fonctionner correctement, lorsqu'on doit disposer d'un paquet intégral (pour décompression ou désérialisation).

Le mode transfert complet est implémenté comme une succession de transferts courts, jusqu'au remplissage du buffer de réception. Il évite par conséquent au développeur de rassembler les trames applicatives reçues.

	transfert court	transfert complet
Synchrone	socket::read_some() socket::write_some() socket::receive() socket::send()	boost::asio::read() boost::asio::write()
Asynchrone	socket::async_read_some() socket::async_write_some() socket::async_receive() socket::async_send()	boost::asio::async_read() boost::asio::async_write()

Retrouvez la suite de l'article de Gwenaël Dunand en ligne : Lien58

Donnez une interface à vos script VBS , HTA : Html Application

La question sur la manière de donner une interface au script VBS revient souvent sur le forum ..
Le HTA (Html Application) est l'une des solutions.
Cet article va vous aider à débuter avec hta.

1. C'est quoi le HTA ?

1.1. Présentation

Une Application HTA est constituée d'un fichier texte, avec une extension .HTA. Sa structure est similaire à une page html standard, avec en supplément un tag HTA application :

```
<HTML>
        <HEAD>
                <TITLE>Structure d'une
application HTA</TITLE>
                <HTA:APPLICATION ID = 'AppBase'>
        </HEAD>
        <BODY>
                Une application HTA de BASE
        </BODY>
</HTML>
```

Introduction to HTML Applications (HTAs) (Lien59)

1.2. Exécution d'une application HTA

Les applications HTA sont disponibles sous Windows depuis l'arrivée d'Internet Explorer 4. Les fichiers avec l'extension .HTA sont associés à "Microsoft (R) HTML Application host", mshta.exe. Ils sont donc exécutables par un simple double-clic sur le fichier .HTA considéré.

1.3. Sécurité

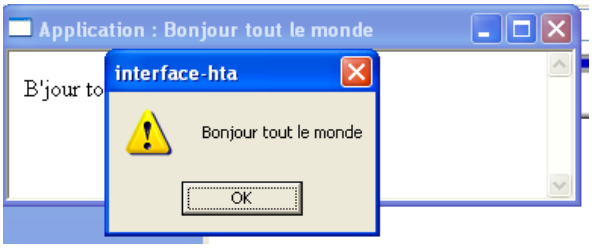
Les Sécurités appliquées aux applications HTA sont un peu différentes de celles appliquées aux pages HTML. Ainsi, une application HTA a accès au système de fichiers (FileSystemObject) , à la base de registre, ainsi que à la technologie WMI.

1.4. L'application "bonjour tout le monde"

Pour ne pas échapper à la tradition de la première application "Hello World", voici un exemple de code, qui permet aussi de voir l'emplacement du code VBScript dans l'entête du fichier.

```
<HTML>
  <HEAD>
    <TITLE>Application : Bonjour tout le
monde</TITLE>
    <HTA:APPLICATION ID = 'AppBonjour'>
      <script language="VBScript">
      Sub Window_onLoad
        msgbox "Bonjour tout le
monde",vbExclamation ,"interface-hta"
      End Sub
    </script>
  </HEAD>
  <BODY>
    B'jour tous.
  </BODY>
</HTML>
```

L'action événementielle "Window_OnLoad" est exécutée au démarrage de l'application.



Application 'Bonjour tout le monde'

Télécharger l'application 'Bonjour tout le monde' (Lien60)

2. Le Tag HTA:APPLICATION

L'application hta est caractérisée grâce à un tag "HTA:APPLICATION" dont les attributs définissent l'apparence et le comportement.

Les propriétés de HTA:APPLICATION

- **ID** : Chaîne de caractères, identifiant de l'application, permettant ensuite au script d'accéder à l'application.
- **APPLICATIONNAME** : Nom de l'application
- **CAPTION** : 'yes' ou 'no' : définit si la barre de titre est affichée ou pas.
- **ICON** : Chemin complet du fichier icône (.ico) 32x32 associé à l'application. L'icône apparaît, l'icône apparaît dans la liste des tâches, la liste des application en cours (ALT-TAB) , en haut à gauche de la fenêtre application...
- **VERSION** : Version de l'application.
- **INNERBORDER** : 'yes' ou 'no' : définit si la bordure intérieure "3D" de la fenêtre est affichée ou pas.
- **MAXIMIZEBUTTON** : 'yes' ou 'no' : affichage du bouton d'agrandissement. Avec 'no' disparaît, ou passe en grisé si le bouton de réduction est présent

- **MINIMIZEBUTTON** : 'yes' ou 'no' : affichage du bouton de réduction. Avec 'no', disparait ou passe en grisé si le bouton d'agrandissement est présent
- **CONTEXTMENU** : 'yes' ou 'no' , affichage du menu contextuel (clic sur la fenêtre, du bouton droit de la souris)
- **NAVIGABLE** : 'yes' ou 'no' , pour yes la navigation s'effectue dans la fenêtre courante, pour no : une nouvelle fenêtre est ouverte lors d'une action sur un lien.
- SCROLL
    - 'yes' : Affichage des barres de défilement
    - 'no' : Les barres de défilement ne sont pas affichées
    - 'auto' : Les barres de défilement apparaissent automatiquement lorsque le contenu de la fenêtre le réclame.
- **SCROLLFLAT** :'yes' ou 'no' : Définit l'aspect visuel des scrollbar , avec pour 'no' aspect 3D et pour 'yes' aspect "plat" (sous XP, le changement d'aspect n'est pas vraiment flagrant..)
- **SELECTION** : 'yes' ou 'no' : à "yes" par défaut, autorise la sélection de texte ... dans l'application HTA.
- **SHOWINTASKBAR** : 'yes' ou 'no' : définit si l'application apparaît ou pas dans la barre des tâches.
- **SINGLEINSTANCE** : 'yes' ou 'no' : à 'yes' n'autorise qu'une seule instance de l'application HTA *(pour que la limitation fonctionne il est nécessaire de renseigner l'attribut **ApplicationName**)*
- **SYSMENU** : 'yes' ou 'no' Définit l'affichage du menu système (lors du clic sur icône en haut à gauche), à no, les boutons 'réduction' et 'agrandissement' de la fenêtre sont aussi masqués.
- BORDER
    - 'thick' : Bord "épais" et fenêtre redimensionnable
    - 'thin' : Bord "fin" et fenêtre non-redimensionnable.
    - 'dialog' : Bord standard fenêtre de dialogue.
    - 'none' : fenêtre sans bords (et sans barre de titre).
- BORDERSTYLE : (définit les bords de la fenêtres, difficile de voir les différences entre les diverses versions ..)
    - 'normal' : valeur par défaut, bordure standard
    - 'complex' : bord de la fenêtre avec effet 3D plus accentué.
    - 'raised' : bord de la fenêtre avec un effet 3D moins accentué.
    - 'static' : bord de la fenêtre effet 3D plutôt fin
    - 'sunken' : bord 3D épais
- WindowState
    - 'normal' : Défaut, taille par défaut des fenêtres d'Internet explorer.
    - 'minimize' : Application minimisée, dans barre de tâches.
    - 'maximize' : Application "plein écran".

Je ne vais pas vous détailler ici l'utilisation de toutes ces propriétés. Je vous laisse les découvrir, sachant que les valeurs soulignées sont les valeurs utilisées par défaut pour la propriété considérée.

3. Mettez des contrôles dans vos applications

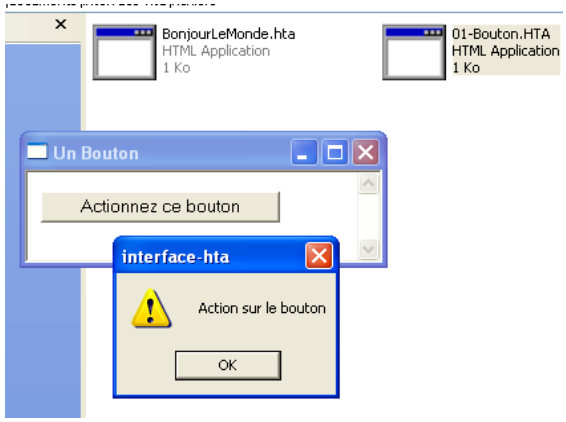
L'ajout de contrôles dans vos applications .HTA, s'effectue de la même manière que pour les pages HTML.

3.1. Ajout d'un bouton

Pour lier le bouton au script associé on peut utiliser la propriété "onClick" du bouton.

```
<HEAD>
<TITLE>Un Bouton</TITLE>
<HTA:APPLICATION
      APPLICATION="htaBouton"

 >
 </HEAD>
 <script language="VBScript">
      Sub tstBouton
           MsgBox "Action sur le
bouton",vbExclamation,"interface-hta"
      End Sub
 </SCRIPT>
 <BODY>
 <input type="button" value="Actionnez ce bouton"
onClick="tstBouton">
 </BODY>
```



Cependant, je préfère utiliser une autre façon pour lier le bouton au script, en créant un fonction événementielle associée au clic sur le bouton.

```
<head>
<title>Un 2' Bouton</title>
<HTA:APPLICATION
      APPLICATION="htaBouton"

 >
 </head>
 <script language="VBScript">
      Sub tstBouton_OnClick
           MsgBox "Action sur le
bouton",vbExclamation,"interface-hta"
      End Sub
 </script>
 <body>
```

```
 <input type="button" value="Actionnez ce bouton"
name="tstBouton">
 </body>
```

Télécharger les 'applications Bouton' (Lien61)

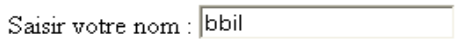
3.2. Un champ de saisie (textbox)

Pour disposer d'un champ de saisie, inspirez-vous de ce code :

```
Saisir votre nom : <INPUT TYPE="text"
NAME="txtNom" SIZE="20" MAXLENGTH="30"
VALUE="bbil">
```

avec **txtNom** comme nom du contrôle, donnant ensuite au code VBScript l'accès à la valeur saisie grâce à sa propriété .value:

```
MsgBox "Votre Nom est : " & txtNom.value
```



Dans le cas de saisie de mot de passe, il est possible de rendre invisible les caractères saisis en modifiant le type de "text" à "password"

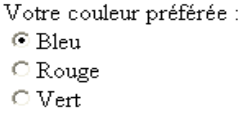
```
<INPUT TYPE="password" NAME="txtMotDePasse"
TITLE="Saisir votre mot de passe" SIZE="20"
MAXLENGTH="30" VALUE="">
```

L'attribut TITLE permet de définir un Tooltiptext, chaîne de caractères qui apparaît dans la bulle d'information lorsque l'utilisateur passe le curseur de la souris sur le boutton.

3.3. Boutons radio

Le type radio permet de donner le choix entre plusieurs propositions (un seul choix possible) :

```
 Votre couleur préférée : <BR>
<INPUT TYPE="radio" NAME="optCouleur"
VALUE="Bleu" CHECKED>Bleu<BR>
<INPUT TYPE="radio" NAME="optCouleur"
VALUE="Rouge">Rouge<BR>
<INPUT TYPE="radio" NAME="optCouleur"
VALUE="Vert">Vert<BR>
<BR>
```



Ensuite l'on teste la propriété **checked** de chacun des éléments du groupe de case à cocher :

```
if optCouleur(0).Checked  then  msgbox "Couleur
choisie : bleu"
if optCouleur(1).Checked  then  msgbox "Couleur
choisie : Rouge"
if optCouleur(2).Checked  then  msgbox "Couleur
choisie : Verte"
```

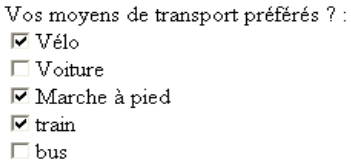
On peut aussi utiliser une bouche for, grâce à la propriété **length** qui renvoie le nombre de cases à option dans le groupe, et à la propriété **value** :

```
Dim stCouleur
Dim i
for i = 0 to optCouleur.length -1
        if optCouleur(i).Checked then
                stCouleur = optCouleur(i).value
                exit for
        end if
next
```

3.4. Les Checkbox

Les checkbox s'utilisent de la même manières que les radios bouton. La différence tient dans le fait que plusieurs cases peuvent être cochées simultanément.
Le code pour ajouter les chekbox à l'application :

```
<BR>
Vos moyens de transport préférés ? : <BR>
<INPUT TYPE="checkbox" NAME="chkTransport"
VALUE="Vélo" CHECKED>Vélo<BR>
<INPUT TYPE="checkbox" NAME="chkTransport"
VALUE="Voiture"CHECKED>Voiture<BR>
<INPUT TYPE="checkbox" NAME="chkTransport"
VALUE="Marche">Marche a pied<BR>
<INPUT TYPE="checkbox" NAME="chkTransport"
VALUE="train" >Train<BR>
<INPUT TYPE="checkbox" NAME="chkTransport"
VALUE="bus">Bus
```



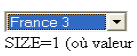
Pour la partie VBScript, on utilise le même principe que pour les radios boutons:

```
For i = 0 to chkTransport.length - 1
        if chkTransport(i).checked then
                stTransport = stTransport &
chkTransport(i).value & ", "
        end if
next
```

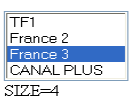
3.5. Liste de choix (donnés fixes)

```
<SELECT NAME="lstChaines" SIZE=4
onChange="ChoixChaine">
<OPTION VALUE="TF1">TF1
<OPTION VALUE="FR2">France 2
<OPTION VALUE="FR3" SELECTED>France 3
<OPTION VALUE="C+" >CANAL PLUS
</SELECT>
```

Le paramètre facultatif SIZE permet de définir le nombre d'éléments de la liste visibles simultanément.



SIZE=1 (où valeur par défaut)



SIZE=4

La propriété : "MULTIPLE" permet d'autoriser les choix multiples dans la liste :

```
 Sports pratiqués : <BR>
 <SELECT NAME="lstSports" SIZE=4 MULTIPLE>
<OPTION VALUE="rugby">rugby
<OPTION VALUE="foot">foot
<OPTION VALUE="basket-ball" SELECTED>Basket
<OPTION VALUE="hand-ball" >Hand
</SELECT>
  <BR><BR>
```

La propriété "Selected" permet d'extraire les éléments choisis de la liste

```
For i = 0 to lstSports.length - 1
        if lstSports(i).Selected then
                stSport = stSport &
lstSports(i).value & " "
        end if
next
```

3.6. Liste de choix de données dynamique

Les éléments compris dans la liste de choix peuvent être définis à l'ouverture de l'application
Dans la partie "Body" du HTA l'on place la déclaration de la liste :

```
<SELECT NAME="lstDyn">
</SELECT>
```

Puis cette liste peut être renseignée par exemple à l'ouverture de l'application :

```
Sub Window_Onload
'Liste box dynamique:
For i = 1 to 10
        Set oOption =
Document.createElement("OPTION")
        oOption.Text = "Opt" & i
        oOption.Value = "Option " & i
        lstDyn.Add(oOption)
next
end sub
```

3.7. Zone de saisie texte multilignes

TEXTAREA permet de définir une zone de saisie :

```
Vos commentaires <BR>
<TEXTAREA NAME="txtCommentaires" ROWS="3"
COLS="40" >
</TEXTAREA>
```

Ses attributs facultatifs sont ROWS pour le nombre de lignes, COLS le nombre de colonnes.
Et READONLY pour un champ à afficher seulement.
L'attribut NAME est utilisé dans le code VBScript pour accéder aux données saisies:

```
Msgbox txtCommentaires.Value ,vbEclamation,"Vos
commentaires"
```

3.8. Sélection d'un fichier

Le tag INPUT de type 'file' permet l'appel de la fenêtre de sélection d'un fichier :

```
<INPUT TYPE="file" NAME="inFichier" SIZE="30"
onChange="choixfichier">
```


```
Sub ChoixFichier
        msgbox inFichier.value
end sub
```

3.9. Télécharger l'application 'Controles'

(Lien62)

3.10. Utiliser un timer

Il ne s'agit pas à proprement parler d'un contrôle, mais de la procédure SetInterval, qui permet de simuler le fonctionnement d'un timer tel qu'on le trouve dans VB6.
Initialisation du timer

```
MonTimer = window.setInterval ("MonScript", 500,
"VBScript") 'Appel de MonScript toutes les 1/2
secondes
```

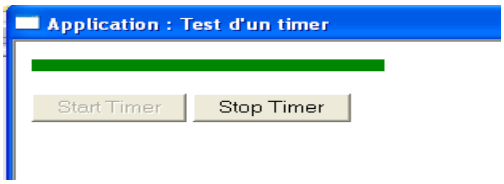
La variable "MonTimer" permet ensuite de Stopper le timer si nécessaire

```
window.ClearInterval MonTimer
```

Un exemple d'utilisation d'un timer :

```
<html>
  <HEAD>
    <TITLE>Application : Test d'un timer</TITLE>
    <HTA:APPLICATION ID = 'AppTimer'>
  <script language="VBScript">
    dim MonTimer
      Sub Window_onLoad
        tb1.width=1
        StartTimer
      End Sub
      Sub StartTimer
        bpTimer.setAttribute "disabled", true
        bpStop.setAttribute "disabled", false
        MonTimer = window.setInterval
("MonScript", 500, "VBScript") 'Appel de
MonScript toutes les 1/2 secondes
      End sub
      sub MonScript
        tb1.width = tb1.width+10
        if tb1.Width > 600 then tb1.Width =1
      End Sub
      Sub bpStop_OnClick
```

```
        bpTimer.setAttribute "disabled", false
        bpStop.setAttribute "disabled", true
        window.ClearInterval MonTimer
      end sub
      </script>
    </HEAD>
    <BODY>
      </TABLE>
<TABLE id="tb1" bgColor=green
 height=10 width=0
 cellSpacing=0 cellPadding=0 border= 0>
 <TR><TD></TD></TR>
</TABLE><BR>
 <input type="button" value="Start Timer"
name="bpTimer" OnClick="StartTimer"
title="Relance le timer">
 <input type="button" value="Stop Timer"
name="bpStop" title="Met fin au timer">
    </BODY>
</html>
```



Télécharger l'application Timer (Lien63)

4. Travailler sur les fichiers

4.1. Introduction au FileSystemObject dans un HTA

Les sécurités relatives au HTA vous permettent d'accéder à la bibliothèque "Scripting Runtime library". Une section de la FAQ VBScript lui est consacrée :
Fichiers : le FileSystemObject (Lien64)

4.2. Un exemple d'utilisation

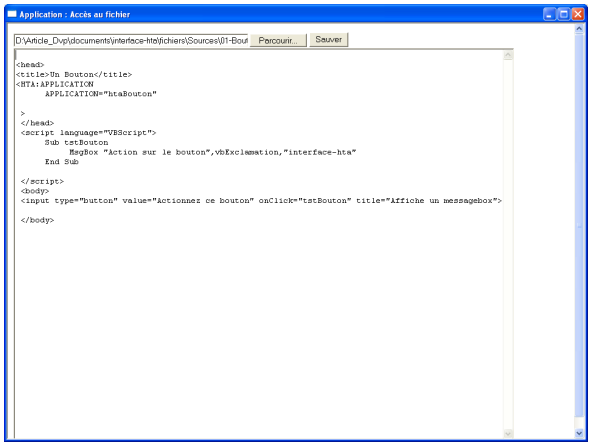
Le code suivant reprend l'exemple standard d'un mini-éditeur de texte :

```
<html>
  <HEAD>
    <TITLE>Application : accès au fichier</TITLE>
    <HTA:APPLICATION ID = 'AppScripting'>
  <script language="VBScript">
Sub ChoixFichier
Const ForReading = 1, ForWriting = 2
Dim oFso, f
Dim stFichier
stfichier = inFichier.value
Set oFso =
CreateObject("Scripting.FileSystemObject")
if oFSO.FileExists(stfichier) then
  Set f = oFso.OpenTextFile(stFichier,
ForReading)
  txtFichier.value = f.ReadALL
  f.Close
else
  msgbox "Fichier " & vbCrlf & stFichier & vbCrlf
& "Inaccessible !",vbCritical,"interface-hta"
end if
  set f = Nothing
  set oFso = Nothing
end sub
'
' Sauvegarde du fichier texte
'
Sub btSauver_OnClick
Const ForReading = 1, ForWriting = 2
Dim oFso, f
Dim stFichier
stfichier = inFichier.value
Set oFso =
CreateObject("Scripting.FileSystemObject")
  Set f = oFso.OpenTextFile(stFichier,
ForWriting)
  f.write txtFichier.value
  f.close
  set f = Nothing
  set oFso = Nothing
end sub

    </script>
  </HEAD>
  <BODY>

  <INPUT TYPE="file" NAME="inFichier" SIZE="60"
onChange="choixfichier">
  <input type="button" value="Sauver"
name="btSauver" >
  <BR>
  <TEXTAREA NAME="txtFichier" ROWS="40"
COLS="100" >
</TEXTAREA>

  </BODY>
</html>
```



Télécharger l'application ScriptingRuntime (Lien65)

5. Accéder à la base de registre

5.1. Introduction : la base de registre en VBS

Les sécurités des applications HTA permettent l'accès à l'objet Wshell et ses trois méthodes destinées à la gestion de la base de registre :

1. **RegWrite** : pour écrire dans la base de registre
2. **RegRead** : pour lire une entrée de la base de registre
3. **RegDelete** : pour effacer une clef ou sa valeur de la base de registre

Pour plus d'informations, consultez la FAQ VBScript :
VBScript : Manipuler la base de registre (Lien66)

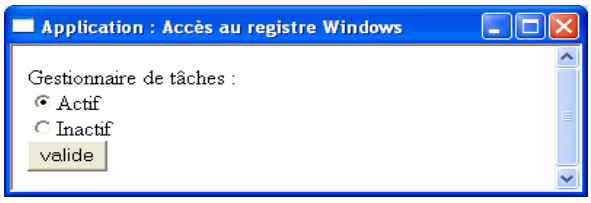
5.2. Un exemple d'utilisation

Pour cet exemple j'ai choisi de donner une interface à une Q/R de la FAQ :
Comment désactiver le gestionnaire des tâches ou la séquence de touches Ctrl-Alt-Suppr ? (Lien67)
Lors de la procédure "Windows_OnLoad" la valeur de la clé est lue et l'état des boutons radios est mis à jour, l'utilisation de la gestion d'erreur permet de traiter le cas ou la clé d'inhibition n'existe pas (par défaut à la première utilisation).

```
<html>
  <HEAD>
    <TITLE>Application : accès au registre
Windows</TITLE>
    <HTA:APPLICATION ID = 'AppRegistre'>
  <script language="VBScript">
    '
    ' A l'ouverture de l'application lecture de
la valeur courante de la clef de registre
concernée
    '
        Const
CLEGTACHE="HKEY_CURRENT_USER\Software\Microsoft\W
indows\CurrentVersion\Policies\System\DisableTask
Mgr"
      Dim WshShell 'variable globale WshShell


      Sub Window_onLoad


        dim stDisable
          stDisable = ""
          Set WshShell =
CreateObject("WScript.Shell")
          On Error resume next ' Si la clef de
registre n'existe pas l'on garde la valeur "".
          stDisable = WshShell.RegRead
(CLEGTACHE)
          On Error Goto 0
          if stDisable = "" then
optGtaches(0).Checked = true else
optGTaches(1).checked = true
      End Sub
      Sub btValider_OnClick
        dim stDisable
          if optGtaches(0).Checked then stDisable
="" else stDisable="1"
          WshShell.RegWrite CLEGTACHE,stDisable
      end sub
    </script>
  </HEAD>
  <BODY>
Gestionnaire de tâches : <BR>
<INPUT TYPE="radio" NAME="optGTaches"
VALUE="Actif" >Actif<BR>
<INPUT TYPE="radio" NAME="optGTaches"
VALUE="Inactif">Inactif<BR>
<input type="button" value="Envoi"
name="btValider" >
  </BODY>
</html>
```



Télécharger l'application Accés au registre (Lien68)

6. Utiliser WMI

6.1. Introduction WMI

Une application HTA peut-être utilisée pour exécuter des requêtes WMI
Voir la FAQ WMI : Windows Management Instrumentation (Lien69)

6.2. Exemple d'utilisation

Un exemple d'utilisation : Lister les partages d'un ordinateur :

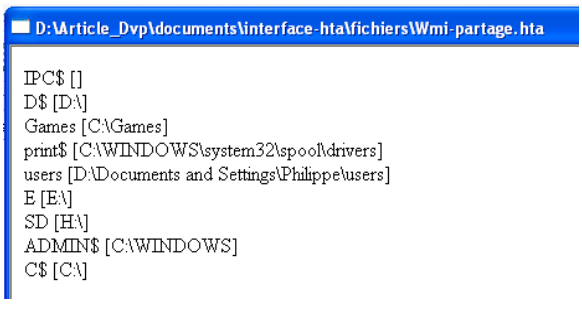
```
<html><HEAD>
                <TITLE>Application :
Partages</TITLE>
                <HTA:APPLICATION ID =
'AppWMIPartages'>
        <script language="VBScript">
Sub Window_onLoad

    Dim strObject
    Dim colShares
    Dim objWMIService, objShare

    Set objWMIService = GetObject( "winmgmts:" )
    Set colShares =
objWMIService.ExecQuery( "Select * from
Win32_Share" )
    For Each objShare In colShares
        Document.Write objShare.Name & " [" &
objShare.Path & "]<BR/>"
    Next

End Sub
</script>
        </HEAD>
        <BODY>


        </BODY>
</html>
```



Télécharger l'application exemple WMI (Lien70)

Retrouvez la suite de l'article de bbil en ligne : Lien71

Comment l'optimiseur d'Oracle calcule le coût

L'analyse des plans d'exécution des requêtes SQL met en évidence des valeurs numériques affichées dans des colonnes appelées Cost (coût) et Rows (lignes) pour chaque opération effectuée par l'optimiseur. S'il est évident que le meilleur plan devrait être celui qui a le moindre coût, les algorithmes utilisés ne sont pas toujours détaillés dans la documentation d'Oracle. Pourtant ils ont fait depuis des années l'objet des d'analyses minutieuses pour comprendre leur fonctionnement et les hypothèses sur lesquels ils sont basés.

1. Introduction

SQL est un langage non procédural et cela signifie que l'utilisateur du langage, c'est-à-dire le développeur des applications, spécifie en fait quelles données chercher et non pas l'algorithme exact à appliquer pour ramener ces données. Trouver le plan d'exécution d'une requête, c'est-à-dire la méthode la plus efficace pour exécuter la requête SQL est le rôle d'un composant logicielle appelée optimiseur. Au départ dans les premières versions d'Oracle l'approche prise dans le développement d'un optimiseur a été assez simpliste mais, remarquablement efficace pour l'état des choses de l'époque, inventorier les méthodes d'accès aux données disponibles : balayage de la table, utilisation d'un index, utilisation d'un pointer (le Rowid), etc. et les classifier par ordre d'efficacité. Ainsi le plus efficace est d'accéder aux données par Rowid (pointer) et le moins par balayage complet de la table. Des autres règles concerne les algorithmes des jointures notamment le choix de la table directrice. Sans entrer dans les détails il est facile à comprendre que cet optimiseur est totalement stable : ses choix ne dépendent pas de la volumétrie des données mais, plutôt de la disponibilité ou pas de certaines méthodes d'accès, en spécial les indexes. C'était une solution honorable qui fonctionnait bien dans pas mal des cases mais, qui était aussi réputée pour générer de mauvais plans d'exécution surtout si plusieurs indexes composite pouvait se qualifier pour l'exécution de la requête et de prendre la mauvaise table comme table directrice dans les jointures. A part l'intéresse historique qui permet de comprendre la raison des quelques vieilles conseils d'optimisation difficilement justifiable dans l'état actuel des choses cet optimiseur ne présent plus d'intérêt, notamment parce que les nouvelles méthodes d'accès ou de jointures ne lui sont pas disponibles. Dans la suite seulement l'optimiseur basé sur le coût est analysé.

1.1. Composants de l'optimiseur

L'optimiseur est constitué de trois composants qui s'enchaînent les entrée-sorties

- **Le transformateur de requête** qui prend en entrée une requête parsée et il essaie de la transformer de telle façon qu'un meilleur plan d'exécution peut être généré, en employant des techniques de transformation de type : fusionner la requête avec une éventuelle vue (view merging), le déplacement des prédicats (predicate pushing), transformation des sous-interrogations en jointures (unnesting sub queries), etc.
- **L'estimateur** qui pour la requête transformée génère trois types de mesure : sélectivité, cardinalité et coût.
- **Le générateur de plan** qui explore plusieurs possibilités d'exécution de la requête et prends celle qui a le meilleur coût.

Le but de cet article est de donner un aperçu des algorithmes employés par l'estimateur pour calculer la sélectivité, les cardinalités et le coût en explorant quelques différences entre les versions 9i, 10g et 11 d'Oracle. L'optimiseur est une composante logicielle extrêmement complexe et qui connaît des améliorations et modifications constantes d'une version à l'autre. Donner les détails précis de son fonctionnement dans diverses situation est une oeuvre assez complexe qui dépasse largement le cadre de ce tutoriel. Le lecteur intéressé par ce sujet est invité à étudier ce livre (Lien72)

1.2. Sélectivité, cardinalité, coût

La sélectivité représente une partie des lignes d'un ensemble des lignes (tables, vue, résultat d'une jointure ou d'un opérateur GROUP BY). La notion de sélectivité est liée à la notion de prédicat ou de combinaison des prédicats de la requête comme par exemple dans « colonne = valeur ». La sélectivité d'un prédicat indique combien de lignes d'un ensemble des lignes passent le test de filtrage du prédicat. Les valeurs de la sélectivité sont dans le domaine 0.0 à 1.0 où 0.0 signifie qu'aucun ligne ne sera pas sélectionnée et 1.0 que toutes les lignes seront sélectionnées. Le calcul de la sélectivité est basée sur les statistiques des objets à accéder. Si ces statistiques ne sont pas disponibles l'estimateur utilise pour ses calculs des valeurs implicites en Oracle9 et en plus l'échantillonnage dynamique à partir d'Oracle 10g (Le paramètre qui contrôle l'échantillonnage dynamique est présent déjà en Oracle9i mais, sa valeur est 1 par rapport à la valeur 2 en Oracle 10g).

La cardinalité représente le nombre des lignes d'un ensemble des lignes (tables, vue, etc.)

Le coût représente des unités de travail ou des ressources utilisées pour exécuter la requête. Le nombre des opérations d'entrées/sortie de disque, l'utilisation du CPU et de la mémoire représentent des unités de travail. Le

chemin d'accès détermine le nombre des unités de travail nécessaire à l'obtention des données de la table. De cette façon le coût d'accès d'un balayage de la table dépende de nombre des blocs à lire et de la valeur d'un paramètre qui indique le nombre des blocs lus dans une opération de lecture E/S multiblocs. Le coût d'accès via un index dépende de nombre des niveaux d'index, de nombre des blocs feuille qu'il faut balayer et du nombre des lignes ramenées en utilisant le pointer (rowid) associé aux clefs d'index. Le coût d'une jointure est une combinaison des coûts d'accès individuels plus le coût de l'opération de jointure : nested loop, sort merge joint ou hash join.

1.3. Le paramètre OPTIMIZER_MODE

Il y a en fait trois variants de l'optimiseur basé sur le coût identifiés par la valeur du paramètre OPTIMIZER_MODE.

- **ALL_ROWS** : l'optimiseur essaie de trouver le plan d'exécution qui a le meilleur temps de réponse global
- **FIRST_ROWS_N** : l'optimiseur essaie de trouver le plan d'exécution qui permet de ramener le plus rapidement possible les premières N lignes où N prends les valeurs 1, 10, 100 ou 1000.
- **FIRST_ROWS** : l'optimiseur essaie de trouver le plan d'exécution qui permet de ramener le plus rapidement possible le premier enregistrement mais, en utilisant un mixe de coût et heuristique

Le but de chaque variante est toujours le même : trouver le meilleur plan d'exécution, mais du aux règles différentes implémentés dans l'optimiseur le résultat n'est pas identique; par exemple on mode FIRST_ROWS_N l'optimiseur va favoriser l'utilisation d'un index pour une même requête qui en mode ALL_ROWS pourrait être exécutée avec un balayage de la table.

Les autres valeurs possibles : CHOOSE, choix de l'optimiseur coût ou basée sur les règles en fonction de la disponibilité des statistiques pour les tables et indexes utilisées et RULE, optimiseur basée sur les règles sont disponibles pour compatibilité antérieure et ne doivent pas être utilisées normalement à partir d'Oracle 9. Par défaut en Oracle10g, le but de l'optimiseur est le meilleur temps de réponse global.

2. Les relations entre prédicats et sélectivité

Dans les plans d'exécution Oracle fait apparaître les filtres utilisés pour calculer le nombre des lignes ramenées. La théorie de probabilités nous enseigne que si nous jetons un dé et nous ne posons la question de savoir combien de chances il y en a d'obtenir la valeur 6, la réponse est de 1/6 ce qui représente un divisé par le nombre des valeurs distinctes que le dé peut avoir. D'une manière analogue on peut supposer que le nombre des lignes ramène par une requête qu'utilise un prédicat de type "col = 6" pour une table où il y a seulement six valeurs distinctes pour cette colonne, sera de 1/6 multiplié par le nombre des lignes de la table. Cette fraction: un divisé par le nombre des valeurs distinctes de la colonne, est précisément la valeur du facteur de filtrage pour une requête utilisant un prédicat qui implique une seule colonne avec condition d'égalité :

```
FF = 1/NDV = densité
```

où NDV est le nombre des valeurs distinctes de la colonne.

Les relations entre le Facteur de Filtrage et les prédicats sont :

- Sans variable de liaison, sans histogrammes et si les colonnes sont déclarées non nulles (T1)

Prédicat	Facteur de Filtrage
C1 = valeur	c1.density
C1 like valeur	c1.density (*)
C1 > valeur	(Hi - valeur) / (Hi - Low)
C1 >= valeur	(Hi - valeur) / (Hi - Low) + 1 / c1.num_distinct
C1 < valeur	(Valeur - Low) / (Hi - Low)
C1 <= valeur	(Valeur - Low) / (Hi - Low) + 1 / c1.num_distinct
C1 between val1 and val2	(Val2 - Val1) / (Hi - Low) + 2 * 1 / c.num_distinct

- Avec variables de liaison (T2)

Prédicat	Facteur de Filtrage
C1 = :b1	c1.density
C1 {like | < | <= |> | >=} :b1	{ 5.0e-02 | c1.density }
C1 between :b1 and :b2	5.0e-02 * 5.0e-0.2

- La combinaison des prédicats (T3)

Prédicat	Facteur de Filtrage
prédicat1 and prédicat2	FF1 * FF2
prédicat1 or prédicat2	FF1 + FF2 - FF1 * FF2

où density, high_value (Hi), low_value (Low), num_distinct sont des colonnes de la vue (dba|all|user)_tab_columns.

(*) En fait plusieurs formules ont été observées pour ce cas. Col like '%' signifie tous les lignes et l'optimiseur utilise la statistique NumRows. Col like 'A%' est souvent interprété comme col >= 'A' and col < 'B', etc. Avec l'augmentation des caractères de la chaîne littérale les deux formules donnent le même résultat.

3. L'équation de calcul du coût

L'équation utilisée pour estimer le coût est :

```
(1) cost = (#SRDS * sreadtm +
            #MRDS * mreadtm +
            #CPUCycles/cpuspeed) / sreadtm
```

où

- **#SRDS** est le nombre des lectures E/S en mode mono bloc
- **sreadtm** est le temps nécessaire à la lecture d'un bloc
- **#MRDS** est le nombre des lectures en mode

multiblocs
- **mreadtm** est le temps de lecture en mode multiblocs
- **CPUCycles** est le nombre des cycle CPU
- **cpuspped** est le nombre des cycle CPU par seconde

cette équation peut être réarrangée pour donner

```
(1a) cost = #SRDS  +
           #MRDS * mreadtm / sreadtm +
           #CPUCycles / (cpuspeed * sreadtm)
```

Le modelé de coût impliqué par l'équation (1) a été introduite en Oracle 9 et il reste valable pour Oracle 10g et 11 quoi que de différences subtiles existent entre les versions. Avant Oracle 9, l'optimiseur ne faisait pas de distinction entre la lecture en mode mono bloc et la lecture en mode multiblocs et ne prenait pas en compte le fait que la lecture des données consomme du temps CPU. En Oracle 8 et avant l'équation se réduisait à sa forme la plus simple

```
(1b) cost = #RDS
```

où RDS inclut les lectures en mode mono bloc et multiblocs sans distinction.

En dépit du fait que l'équation est restée la même le comportement de l'optimiseur a changé d'une version à l'autre. Ainsi, en Oracle8 ou Oracle9i sans le calcul des statistiques système c'est l'ancienne équation (1b) qui est employée. À partir d'Oracle9i une fois que les statistiques système ont été calculées et à partir d'Oracle 10g c'est toujours l'équation (1a) qui est utilisée. En fait en Oracle 10g et 11 les statistiques suivantes sont initialisées du démarrage de la base :

- **ioseektime** : temps de recherche d'un bloc sur le disque en ms, valeur par défaut 10
- **iotfrspeed** : la vitesse de transfert de données en Bytes par ms, valeur par défaut 4096
- **cpuspeedNW** : le nombre moyen de cycles CPU par seconde, valeur par défaut qui dépende du système.
- **MBRC** : le nombre des blocs lus en moyenne dans une opération E/S en mode multiblocs

Les valeurs de ces statistiques sont ensuite utilisées pour calculer les composants nécessaires à l'équation de coût :

```
    sreadtm  = ioseektm + db_block_size /
iotfrspeed
(2) mreadtm  = ioseektm + MBRC * db_block_size /
iotfrspeed
    cpuspeed = cpuspeedNW
```

4. Étude de cas : balayage complet de la table

4.1. Calcul du coût

Le jeu d'essai proposé est le suivant :

```
DROP TABLE bigemp
/
DROP TABLE bigdept
/
CREATE TABLE bigemp (
  empno PRIMARY KEY, ename, job, mgr, hiredate,
sal, comm, deptno
)
pctfree 99
pctused 1
AS
SELECT empno+x AS empno, ename, job, mgr,
hiredate, sal, comm, deptno+x AS      deptno
  FROM emp
CROSS JOIN (SELECT level*1000 AS x FROM dual
CONNECT BY level <= 1000)
/
CREATE TABLE bigdept (
  deptno PRIMARY KEY, dname, loc
)
pctfree 99
pctused 1
AS
SELECT deptno+x AS deptno, dname, loc
  FROM dept
CROSS JOIN (SELECT level*1000 AS x FROM dual
CONNECT BY level <= 1000)
/
```

les statistiques étant calculées de la manière suivante :

```
exec dbms_stats.gather_table_stats(user,
'BIGEMP', cascade=>true)
exec dbms_stats.gather_table_stats(user,
'BIGDEPT', cascade=>true)
```

Quelques remarques à faire : comme pré requis le jeu d'essai se base sur des tables et les données d'un ancien jeu d'essaie d'Oracle crées par le script demobld.sql (Lien73). Il y a au départ 14 enregistrements dans la table emp et 4 dans la table dept. Les tables BIGEMP et BIGDEPT contient donc 1000 fois plus d'enregistrements. Les tables du jeux d'essai déterminent l'allocation d'un nombre significatif des blocs du aux valeurs inhabituels des paramètres pctfree et pctused employés.

Les autres paramètres sont :

- **optimizer_mode**=ALL_ROWS
- **optimiser_dynamic sampling** = 1 (Oracle9i) et 2 (Oracle 10g et 11)
- **optimizer_index_caching** = 0
- **optimizer_index_cost_adj** = 100
- **db_block_size** = 8
- **db_file_multiblock_read_count** = 32 (Oracle9i) et 128 (Oracle 10 et 11)

Au départ les statistiques système ne sont pas calculées.

Nous allons commencer avec une base Oracle 9.2 et les requêtes suivantes :

```
SQL> truncate table plan_table;

Table tronquée.

SQL> explain plan set statement_id='BIGEEMPaSsF'
for Select * From bigemp;

Explicité.
```

```
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------


--------------------------------------------
| Id | Operation | Name | Rows | Bytes | Cost |
--------------------------------------------
| 0 | SELECT STATEMENT |  | 14000 | 546K| 864 |
| 1 | TABLE ACCESS FULL | BIGEMP | 14000 |
546K| 864 |
--------------------------------------------

Note: cpu costing is off

9 ligne(s) sélectionnée(s).
SQL> explain plan set statement_id='BIGDEPTaSsF'
for Select * From bigdept;

Explicité.

SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------


--------------------------------------------
| Id | Operation | Name | Rows | Bytes | Cost |
--------------------------------------------
| 0 | SELECT STATEMENT |  | 4000 | 84000 | 126 |
| 1 | TABLE ACCESS FULL | BIGDEPT | 4000 | 84000
| 126 |
--------------------------------------------

Note: cpu costing is off

9 ligne(s) sélectionnée(s).

SQL> Select table_name, num_rows, blocks,
avg_row_len
  From user_tables
 Where table_name in ('BIGEMP','BIGDEPT');
  2    3
TABLE_NAME    NUM_ROWS     BLOCKS  AVG_ROW_LEN
------------ ---------- ---------- -----------
BIGDEPT         4000       2040          21
BIGEMP         14000      14159          40
```

Première chose à remarquer est la note d'en bas de l'affichage du plan d'exécution, concernant la non-prise en compte du modelé de coût associé à l'équation (1). Donc dans le calcul du coût c'est l'équation (1b) qui est utilisée.

```
(1b) cost = #RDS
```

Il est évident que la valeur affichée dans la zone Rows du plan d'exécution provient de la colonne Num_Rows de la vue user_tables. En examinant les colonnes Blocks de la vue user_tables et Cost du plan d'exécution on retrouve le fait que le coût dépende des nombres de blocs lus. En fait la formule employée est

```
(1c) cost = #RDS = Blocs / K + 1
```

où K dépende du paramètre db_file_multiblock_read_count. Dans ce cas db_file_multiblock_read_count a la valeur 32 et donc K prends la valeur 16,41, la valeur du facteur K étant calculé d'une manière empirique.

Faisons les calculs :

Pour BIGEMP le nombre des blocs est 14159

```
cost = 14159 / 16.41 = 862,82
```

Si on arrondie à l'entier immédiatement supérieur : 863 et on ajoute 1 on retrouve le coût 864. La valeur 1 corresponde au paramètre caché _table_scan_cost_plus_one qui a la valeur vraie en Oracle9 et faux en Oracle 8 (en conséquence la même requête en Oracle8 aura comme coût 863).

La table BIGDEPT

```
cost = 2040 / 16,41 = 124,31 donc arrondi 125 + 1
= 126
```

Pour une valeur de paramètre db_file_multiblock_read_count égal à 16, K prends la valeur 10.40 et donc le coût de la requête sur la table BIGEMP devrait être :

```
cost = 14159 / 10,40 = 1361,44 donc arrondi 1362
+ 1 = 1363
```

Faisons la vérification :

```
SQL> alter session set
db_file_multiblock_read_count=16;

Session modifiée.

SQL> explain plan set statement_id='BIGEMP16' for
select * from bigemp;

Explicité.

SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------


--------------------------------------------
| Id | Operation | Name | Rows | Bytes | Cost |
--------------------------------------------
| 0 | SELECT STATEMENT |  | 14000 | 546K| 1363 |
| 1 | TABLE ACCESS FULL | BIGEMP | 14000 |
546K| 1363 |
--------------------------------------------

Note: cpu costing is off

9 ligne(s) sélectionnée(s).
```

Il est intéressant d'examiner le contenu de la table plan_table à ce moment :

```
SQL> select statement_id, id, cost, cpu_cost,
io_cost
  2  from plan_table;

STATEMENT_ID     ID   COST  CPU_COST  IO_COST
-------------- ----- ----- --------- ----------
BIGEEMPaSsF      0    864               864
```

```
BIGEEMPaSsF      1    864               864
BIGDEPTaSsF      0    126               126
BIGDEPTaSsF      1    126               126
BIGEMP16         0   1363              1363
BIGEMP16         1   1363              1363

6 ligne(s) sélectionnée(s).
```

Remarquons que la colonne CPU_COST est vide ce qui met en évidence que seulement le nombre des lectures E/S, colonne IO_COST, contribuent à la constitution du coût.

Il est également utile d'examiner cette même requête avant le calcul des statistiques sur les deux tables :

```
SQL> explain plan set
statement_id='BIGEMPsansStat' for Select * From
bigemp;

Explicité.

SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
------------------------------------------------


------------------------------------------------
| Id | Operation | Name | Rows | Bytes | Cost |
------------------------------------------------
| 0 | SELECT STATEMENT |  | 1156K | 95M|  864 |
| 1 | TABLE ACCESS FULL | BIGEMP | 1156K| 95M|
864 |
------------------------------------------------

Note: cpu costing is off

9 ligne(s) sélectionnée(s).
```

Le coût est le même en dépit du fait que la colonne Blocks de la vue user_tables est nulle à ce moment mais, le mystère s'explique peut être en interrogeant la colonne Blocks de la vue user_segments. Paradoxalement c'est en absence des statistiques que l'estimation du nombre des blocs est la plus proche de réalité si on prend en compte que les statistiques peuvent être périmées. Par contre, le nombre des lignes ramènees, c'est-à-dire la cardinalité, est bien loin de la réalité, étant presque 100 fois plus importante, sans doute du au nombre important des blocs qui composent les 2 tables. Et c'est bien ce type d'erreur d'estimation qui conduit très souvent à un mauvais plan d'exécution. C'est donc la raison principale d'un conseil d'optimisation qui demande de s'assurer avant toute action que les statistiques pour les objets accédés sont à jour, surtout pour une base Oracle9.

À partir de la version 10g, les choses se sont nettement améliorées :

```
SQL> explain plan set statement_id='BIGEEMPsSsF'
for Select * From bigemp;

Explicité.

SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
------------------------------------------------
```

```
Plan hash value: 2368838273

-----------------------------------------------
| Id | Operation | Name | Rows | Bytes | Cost
(%CPU)| Time |
-----------------------------------------------
| 0| SELECT STATEMENT | | 13934| 1183K| 3853 (1)|
00:00:47 |
| 1| TABLE ACCESS FULL| BIGEMP | 13934 | 1183K|
3853 (1)| 00:00:47 |
-----------------------------------------------

Note
-----

PLAN_TABLE_OUTPUT
--------- -------------------------------------


   - dynamic sampling used for this statement

12 ligne(s) sélectionnée(s)
```

l'explication étant donnée par la remarque d'en bas du plan d'exécution : l'échantillonnage dynamique. En fait le paramètre optimizer_dynamic_sampling a la valeur 1 comme valeur par défaut en Oracle 9 mais, 2 en Oracle 10 ce qui signifie en effet l'application d'une politique d'échantillonnage plus agressive. Remarquez comme le nombre de lignes estimées 13934 est assez proche de la réalité 14000. Mais bien sûr, cette amélioration implique des temps plus longs d'élaboration des plans d'exécutions, donc la recommandation de s'assurer que les statistiques des segments sont à jour reste toujours valable.

Nous allons maintenant activer les statistiques système en utilisant les procédures du package dbms_stat. Au lieu de collecter ces statistiques nous allons mettre à jour ces statistiques avec les valeurs suivantes :

- cpuspeed, le nombre des cycles CPU par seconde: 500
- sreadtim, le temps de lecture en mode mono-block: 5 ms
- mreadtim, le temps de lecture en mode multi-block: 30 ms
- MBRC, le nombre des block lus en mode multi-block : 16

```
SQL> Begin
  DBMS_STATS.SET_SYSTEM_STATS('cpuspeed',500);
  DBMS_STATS.SET_SYSTEM_STATS('sreadtim',5.0);
  DBMS_STATS.SET_SYSTEM_STATS('mreadtim',30.0);
  DBMS_STATS.SET_SYSTEM_STATS('mbrc',16);
end;
/

Procédure PL/SQL terminée avec succès.
```

Affichons ensuite le contenu de la table de statistiques système (Oracle recommande l'utilisation de la procédure dbms_stats.get_system_stats) :

```
SQL> r
  1  select sname, pname, pval1
  2  from sys.aux_stats$
```

```
  3* where sname = 'SYSSTATS_MAIN'

SNAME           PNAME                PVAL1
--------------- --------------- ----------
SYSSTATS_MAIN   CPUSPEED                500
SYSSTATS_MAIN   MAXTHR                   -1
SYSSTATS_MAIN   MBRC                     16
SYSSTATS_MAIN   MREADTIM                 30
SYSSTATS_MAIN   SLAVETHR                 -1
SYSSTATS_MAIN   SREADTIM                  5

6 ligne(s) sélectionnée(s).
```

Et nous allons ignorer dans la suite MAXTHR et SLAVETHR. (Pour supprimer ces statistiques vous pouvez utiliser la méthode brutale et probablement non recommandée qui consiste à supprimer le contenu de la table sys.aux_stats$. À partir d'Oracle10g il est possible de restaurer les valeurs antérieures stockées dans la table d'historique SYS.WRI_OPTSTAT_AUX_HISTORY via le package dbms_stats)

Examinons actuellement la même requête sur la table BIGEMP

```
SQL> explain plan set
statement_id='BIGEMPsysStat' for select * from
bigemp;

Explicité.

SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
------------------------------------------------


------------------------------------------------
| Id | Operation | Name | Rows | Bytes | Cost
(%CPU)|
------------------------------------------------
| 0 | SELECT STATEMENT | | 14000| 546K| 5353 (1)|
| 1 | TABLE ACCESS FULL | BIGEMP | 14000 |
546K| 5353 (1)|
------------------------------------------------

7 ligne(s) sélectionnée(s).

SQL> select id, cost, cpu_cost, io_cost
  2  from plan_table
  3  where statement_id = 'BIGEMPsysStat';
```

```
        ID       COST   CPU_COST    IO_COST
---------- ---------- ---------- ----------
         0       5353  104892469       5311
         1       5353  104892469       5311
```

Naturellement la note indiquant que le modelé de coût CPU n'est pas actif a disparu et le coût a changé d'une manière radicale. En fait c'est l'équation (1a)

```
(1a) cost = #SRDS  +
            #MRDS * mreadtm / sreadtm +
            #CPUCycles / (cpuspeed * sreadtm)
```

qui est utilisée pour le calculer de la manière suivante :

- #SRDS est égal à 0 (balayage complet de la table)
- #MRDS = Blocks / MBRC
- mreadtm = 30
- sreadtm = 5
- cpuspeed = 500
- #CPUCycles = colonne cpu_cost dans plan_table

et donc pour la partie E/S

```
io_cost = #MRDS * mreadtm / sreadtm
io_cost = (14159/ 16) * 30/5 = 884,9375 * 6 =
5309, 625
```

et après arrondi à l'entier supérieur et ajout de 1 (_table_scan_cost_plus_one égal true)

```
io_cost = 5311
```

Pour la partie cpu_cost

```
cpu_cost = #CPUCycles / (cpuspeed * sreadtm)
cpu_cost = 104892469/(500 000 * 5) = 41.95 = 42
après arrondi
```

Et finalement le coût est

```
cost = cpu_cost + io_cost = 5311 + 42 = 5353
```

À partir d'Oracle 9 une fois que les statistiques système sont calculées le coût d'exécution de la requête fait la différence entre les lectures en mode monobloc et multiblocs et démontre que l'utilisation du CPU a une importance dans le calcul du coût.

Retrouvez la suite de l'article de Marius Nitu en ligne : Lien74

Réduire la taille d'un device sous ASE

Théoriquement, il n'est pas possible de réduire la taille d'un device d'une base de données Sybase ASE. Pratiquement, il est nécessaire de devoir le faire parfois, bien que cela ne soit pas officiellement supporté.

1. Introduction

Issue de l'époque lointaine (au moins 10 ans !) ou il n'était pas aisé de faire cohabiter une grosse masse d"information sur de nombreux raw devices, ventilés sur de non moins nombreux disques, la gestion des fichiers de donnée sous Sybase ASE peut paraître quelque peu complexe au néophyte.

Le prix à payer à ce jour est une mauvaise compréhension de la complexité de la gestion des espaces de stockage du DBA Junior, et la mécompréhension de l'inexistence de 2 spécificités existant déjà dans d'autres SGBDR :

1. L'inexistence d'une clause AUTOEXTENT lors de la création de devices (cf. autre article du même auteur (Lien75)
2. L'incapacité de réduire la taille d'un device ayant été trop largement taillé

Cet article a pour but d'apporter une solution à la soit disant impossibilité de réduire un devices sous ASE.

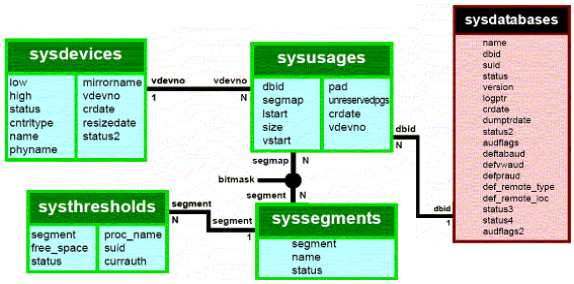
2. L'explication historique

Sybase ASE (anciennement Sybase SQL Server) permettait en fait de définir avant tout ses espaces de stockage, puis ensuite de tailler dans lesdits espaces les segments nécessaires à chaque base de données.

Il faut garder en mémoire que jusqu'en version 11.9, le seul système de stockage supporté en production était le raw device. Il permettait d'assurer une intégrité des accès disques à ASE (en évitant tout cache OS). Ce raw device était généralement créé par l'administrateur système, et non pas par l'administrateur de la base. Une fois cet espace créé, sa taille ne pouvait varier... ce qui explique que l'autoextension ou la réduction du fichier ne pouvait être une option au niveau de l'instance ASE.

Nous avions donc une application claire de ce que Codd souhaitait : une dissociation claire et nette de la partie logique (base de données, tables) et de la partie physique (devices, fichiers sur disques, ...).

Ce qui différencie donc ASE de la plupart de ses concurrents, et ce qui explique certaines contraintes, c'est que l'on y crée des devices neutres. Ce n'est qu'à l'affection de zones que l'on détermine leur contenu, et que le contenu peut être variable. C'est d'ailleurs pour cela que le stockage des informations sur les devices se fait dans la table *sysdevices* de master, puis que c'est ensuite une table de liens *master..sysusages* qui stocke le partages des devices de *sysdevices* avec les dbid de *sysdatabases*.



sysusages

Microsoft a cassé cette logique dès la version 2000 en rapatriant les informations de stockage dans la base elle-même, et en dissociant fortement les fichiers de données (.mdb) et ceux du journal de transactions (.ldb).

Afin d'éviter d'éventuels verrouillages et des accès disques très prétéritants sur les tables système, le métamodèle de Sybase ASE a toujours tenté de rester le plus normalisé possible. Pour des raisons de performances, l'engineering de Sybase s'est toujours fait un devoir de minimiser les accès sur ces tables. Une insertion dans une table, tant qu'elle ne modifie pas un index, ne provoquera quasi pas d'écriture dans le métamodèle.

C'est pour cette raison principale que vous ne trouverez pas, comme chez Oracle avec sa vue *DBA_SEGMENTS*, une liste référençant l'adresses de toutes les pages allouées pour un objet. Sous ASE, tout est chaînage de pages. Dans les tables systèmes, et principalement dans sysindexes, ne sont stockées que quelques informations concernant l'adressage du début du chaînage. Si l'on souhaite obtenir des informations sur le chaînage complet d'un objet, ce sont des outils lents et gourmands tels que dbcc page, dbcc pglinkage , ... qu'il faut alors utiliser.

Cet état de fait est la raison pour laquelle le rétrécissement d'un segment, et de fait d'un device, n'est pas quelque chose de supporté sous ASE.

Pour réduire un device en dessus de ce qui lui est alloué, il "suffit" de mettre sa valeur high à la valeur souhaitée. Si des segments lui sont déjà attribués, il faudra avant tout aller vider ces segments... ou s'assurer qu'ils soient vides... avant de corriger dans sysusages

3. Réduction de la taille d'un device

Comme je l'ai spécifié plus haut, c'est la table master..sysusages que nous allons devoir traiter ici.

Il faut envisager, pour un dbid spécifique et par ordre de lstart, les couches de segments comme des couches d'un sandwich.

Instructions SQL

```
select * from sysusages where dbid=db_id('DVP')
```

Résultats

	dbid	segmap	lstart	size	vstart	pad	unreservedpgs	crdate	vdevno
1	6	3	0	51200	0	(NULL)	50156	2009-01-09 14:14:19.423	13
2	6	4	51200	25600	0	(NULL)	3195	2009-01-09 14:14:19.423	15
3	6	8	76800	51200	0	(NULL)	51000	2009-01-09 14:14:29.423	14
4	6	4	128000	5120	25600	(NULL)	0	2009-01-09 14:28:59.8	15
5	6	3	133120	51200	51200	(NULL)	51000	2009-01-09 14:31:52.153	13
6	6	3	184320	51200	102400	(NULL)	51000	2009-01-09 14:32:04.153	13
7	6	4	235520	20480	30720	(NULL)	19014	2009-01-09 15:25:31.593	15

1. Il est possible de compacter 2 couches conjointes de même nature
2. Il est possible de supprimer une couche supérieure du sandwich si elle ne contient rien
3. Il n'est pas possible de modifier une couche ayant hébergé un segment system

Comme vous pouvez le voir plus haut dans les tables systèmes, depuis la version 15, le lien entre *sysusages* et *sysdevices* ne se fait plus sur la liaison *vstart between low and high*, mais directement sur une clé *vdevno* (le low étant à 0). Selon votre version, vous aurez donc besoin de corriger ce qui suit en conséquent.

La procédure stockée *sp_helpdb* ne nous renseigne que partiellement puisqu'elle ne fait le distinguo qu'entre data et log segments.

```
sp_helpdb DVP
```

Résultats

	device_fragments	size	usage	created	free kbytes
1	DVP_D01	100.0 MB	données seules	jan 9 2009 2:14PM	100312
2	DVP_L01	50.0 MB	journal seulement	jan 9 2009 2:14PM	pas applicable
3	DVP_I01	100.0 MB	données seules	jan 9 2009 2:14PM	102000
4	DVP_L01	10.0 MB	journal seulement	jan 9 2009 2:28PM	pas applicable
5	DVP_D01	100.0 MB	données seules	jan 9 2009 2:31PM	102000
6	DVP_D01	100.0 MB	données seules	jan 9 2009 2:32PM	102000
7	DVP_L01	40.0 MB	journal seulement	jan 9 2009 3:25PM	pas applicable

Afin de déterminer le contenu en segments d'un fragment de fichier, il faut se baser sur la colonne master..sysusages.segmap qui est un masque de bits (Lien76) et sur la table syssegments de chaque base.

```
select u.lstart Start_logique, u.size Taille, u.vstart Start_virtuel, d.name Device, s.name Segment
from sysusages u
inner join sysdevices d on d.vdevno=u.vdevno
inner join DVP..syssegments s on segmap&power(2,segment)=power(2,segment)
where dbid=db_id('DVP')
order by lstart
```

Résultats

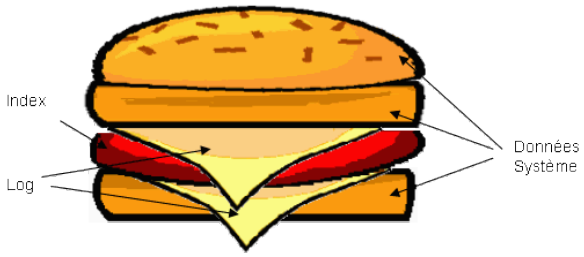
	Start_logique	Taille	Start_virtuel	Device	Segment
1	0	51200	0	DVP_D01	system
2	0	51200	0	DVP_D01	default
3	51200	25600	0	DVP_L01	logsegment
4	76800	51200	0	DVP_I01	indexsegment
5	128000	5120	25600	DVP_L01	logsegment
6	133120	51200	51200	DVP_D01	system
7	133120	51200	51200	DVP_D01	default
8	184320	51200	102400	DVP_D01	system
9	184320	51200	102400	DVP_D01	default
10	235520	20480	30720	DVP_L01	logsegment

Notons la clause de jointure entre syssegments et sysusages qui permet de détecter quel segment se trouve sur quel fragment de device.

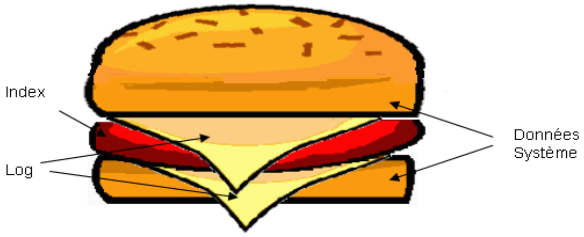
Relevons que par choix, nous avons laissé le segment system avec le segment data... puisque nous n'utilisons qu'un seul device spécifique pour ces 2 types de segments.

Notons enfin que la représentation tabulaire par segment pourrait nous induire en erreur puisque les 2 premières lignes ne représentent en fait qu'un device comprenant 2 segments.

Faisons fi de la couche la plus haute de journal et imaginons-nous ces fameuses couches en sandwich. Nous aurions donc:



Dans le premier cas, il est aisé de coupler 2 segments de même nature puisque nous sommes en mesure de déterminer très clairement qu'ils appartiennent au même device et qu'ils sont conjoints



En effet, l'adresse de start logique est dans notre exemple à 133120 et la taille du segment est de 51200. Le départ de l'adresse logique suivante est à 184320 = 133120+51200. On est donc certain qu'aucun segment intercalaire n'existe.

On peut donc sans autre effectuer la modification suivante directement dans la sysusages, ceci étant bien entendu totalement non supporté officiellement par Sybase.

```
use master
exec sp_configure "allow update",1
begin tran
-- Modifications
update sysusages set size=102400 where
lstart=133120 and dbid=db_id('DVP')
delete sysusages where lstart = 184320 and
dbid=db_id('DVP')

-- Contrôle
select * from sysusages where dbid=db_id('DVP')

-- Validation (ou rollback au cas où !)
commit
exec sp_configure "allow update",0

use DVP
dbcc traceon(3601)
dbcc checkcatalog
select @@error
```

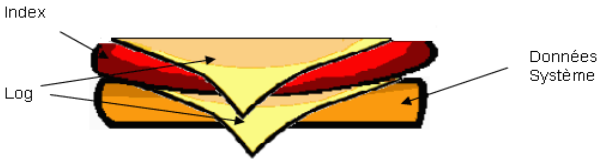
```
Temps d'exécution : 0,031 seconde(s)
Vérification de current database: La taille de page logique est de 2048 octets.
Les segments suivants ont été définis pour la base de données 6 (nom de la base de données DVP).
numéro de device virtuel   adr. virtuelle début   taille (pages logiques)   segments
-------------------------  ----------------------  ------------------------  ------------
13                         0                       51200
                                                                             0
                                                                             1
15                         0                       25600
                                                                             2
14                         0                       51200
                                                                             3
15                         25600                   5120
                                                                             2
13                         51200                   102400
                                                                             0
                                                                             1
15                         30720                   20480
                                                                             2
Exécution de DBCC terminée. Si DBCC a imprimé des messages d'erreur, contactez un utilisateur exerçant les fonctions d'administrateur système (SA).
```

Ce traitement n'as pas réellement de plus-value, hormis de prouver sa faisabilité et de supprimer la trace du disk remap.

Le chaînage des tailles et des adresses logiques explique aussi pourquoi il n'est pas aisé de réduire la taille d'un device. On peut "sans autre" (après avoir déterminé qu'aucun objet ne se trouve dans la partie à tronquer) réduire la taille d'un segment, mais il n'est pas possible de réduire la taille du fichier associé via des commandes T-SQL (on peut toujours el faire à ses risques et péril via une petite routine C insérant un EOF à une adresse donnée).

Nous allons maintenant supprimer le segment. Pour ce faire, nous devions être certain que plus aucun objet ne se trouve dans la tranche que nous voulons supprimer.

Commençons par le bout de journal se trouvant tout au sommet de notre sandwich. Pour tronquer un journal, il suffit de s'assurer que toutes les transactions sont fermée, puis de lancer une opération de troncature.




```
select * from master..syslogshold where
dbid=db_id('DVP')
```

Si cette requête ne retourne pas de ligne, c'est qu'il n'y a pas de transactions ouvertes sur la base DVP, soit le cas que nous souhaitons.

Nous pouvons dès lors tronquer le log et supprimer le haut du sandwich, à savoir :

```
use master
dump tran DVP with truncate_only

use DVP
checkpoint

use master
exec sp_configure "allow updates",1
begin tran
-- Modification
delete sysusages
where dbid = db_id('DVP')
and lstart=(select max(lstart) from sysusages
where dbid = db_id('DVP') and segmap=4)

-- contrôle
select * from sysusages where dbid = db_id('DVP')

-- Validation
commit
exec sp_configure "allow updates",0

use DVP
dbcc checkcatalog
select @@error

dbcc checkdb
select @@error
```

Dans le cas d'une base de données répliquée, il y a deux points de troncature. Celui, standard, de la dernière transaction fermée, et celui plus ancien de la dernière transaction fermée et répliquée. c'est cette dernière qui fait foi, mais cela ne devrait pas vous poser problème, le DUMP TRAN refusera de tronquer plus base.

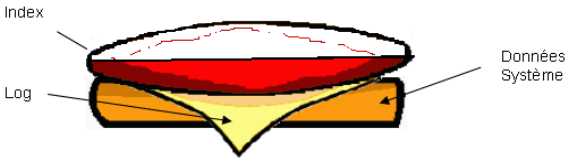
Nous allons attaquer maintenant la réduction du device d'index. Nous nous basons sur ce que nous avons appris auparavant sur 2 blocs contigus de même nature. Nous pouvons donc sans autre couper notre segment d'index en 2 (pour cet exemple, 2 blocs de taille similaire, à savoir 50 Mo chacun). Nous en profitons pour invalider le 2e bloc en lui supprimant tout segment (segmap à 0).

```
use master
exec sp_configure "allow update",1
begin tran
-- Modifications
update sysusages set size=size/2 where
lstart=128000 and dbid=db_id('DVP') -- Division
de la taille par 2, de 100Mo (51200) à 50Mo
(25600)
insert into sysusages values(db_id('DVP'),0,
128000+25600, 25600, 0, null, getdate(), 15) --
Insertion du 2e bloc

-- contrôle
select * from sysusages where dbid = db_id('DVP')

-- Validation
commit

exec sp_configure "allow update",0
```



Que cela soit bien clair : en ce moment, le segment "blanc" du haut n'est plus allouable, mais les pages d'index initiales y sont toujours stockés. Il nous faut donc les déplacer dans la partie vive du bas.

Ceci ne peut se faire via un *sp_placeobject*, cette procédure ne déterminant que l'allocation future pour un objet donné. Il est donc nécessaire de reconstruire les objets du segment.

1. Pour un segment de données, nous passerons donc par un bcp out/bcp in ou un sp_rename + select into OU par la recréation de son index cluster (s'il en existe un)
2. Pour un segment d'index (notre cas précis), par une suppression et une recréation d'index (l'export du DDL pouvant se faire via l'utilitaires ddlgen ou des programmes plus robustes de type PowerAMC (Lien77))

Lorsque le déplacement aura été fait, il nous suffira de supprimer la partie "morte" du segment

```
use master
exec sp_configure "allow update",1
-- Modification
begin tran
delete sysusages where dbid=(db_id('DVP') and
segmap=0

-- contrôle
select * from sysusages where dbid = db_id('DVP')

-- Validation
commit
exec sp_configure "allow update",0
```

Afin de déterminer quel objet se trouve dans le segment en question, vous pouvez sans problème requêter sur colonne segment de la table système *syspartitions* (avant la version 15, sur *sysindexes*).

Vous comprenez sans doute la raison pour laquelle il n'est pas aisé de supprimer un morceau au milieu du sandwich: la partie supprimée laisserait un trou béant difficile à colmater par la suite. Difficile, mais pas impossible : il sera toujours possible de faire coïncider l'adresse de début du bloc mort avec une adresse sommitale d'un autre sandwich = d'une autre base...

4. Conclusion

Comme vous l'avez sans doute remarqué, la procédure est loin d'être triviale. Elle a cependant le mérite d'exister. L'autre option moins élégante mais supportée passe par la recréation d'une base de données et une régénération de tous ses objets. Il reste cependant des cas à forte volumétrie ou à haute disponibilité où cette méthode ne peut malheureusement être applicable.

5. Références

1. Diagramme des tables systèmes (Lien78) de Sybase ASE
2. Documentation (Lien79) de Sybase ASE

Retrouvez l'article de fadace : Lien80

Devices autoextensibles sous ASE

Comment laisser à Sybase ASE le soins de gérer l'augmentation de taille de ses fichiers de données.

1. Introduction

Issue de l'époque lointaine (au moins 10 ans !) ou il n'était pas aisé de faire cohabiter une grosse masse d"information sur de nombreux raw devices, ventilés sur de non moins nombreux disques, la gestion des fichiers de donnée sous Sybase ASE peut paraître quelque peu complexe au néophyte.

Le prix à payer à ce jour est une mauvaise compréhension de la complexité de la gestion des espaces de stockage du DBA Junior, et la mécompréhension de l'inexistence de 2 spécificités existant déjà dans d'autres SGBDR :

1. L'inexistence d'une clause AUTOEXTENT lors de la création de devices
2. L'incapacité de réduire la taille d'un device ayant été trop largement taillé (cf. autre article du même auteur (Lien80))

Cet article à pour but d'apporter une solution à la soit disant impossibilité qu'à ASE d'auto-étendre ses devices.

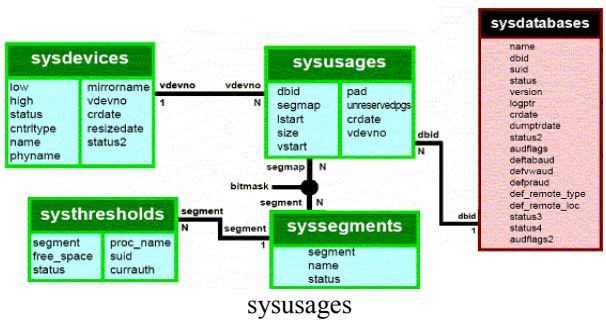
2. L'explication historique

Sybase ASE (anciennement Sybase SQL Server) permettait en fait de définir avant tout ses espaces de stockage, puis ensuite de tailler dans lesdits espaces les segments nécessaires à chaque base de données.

Il faut garder en mémoire que jusqu'en version 11.9, le seul système de stockage supporté en production était le raw device. Il permettait d'assurer une intégrité des accès disques à ASE (en évitant tout cache OS). Ce raw device était généralement créé par l'administrateur système, et non pas par l'administrateur de la base. Une fois cet espace créé, sa taille ne pouvait varier... ce qui explique que l'autoextension ou la réduction du fichier ne pouvait être une option au niveau de l'instance ASE.

Nous avions donc une application claire de ce que Codd souhaitait : une dissociation claire et nette de la partie logique (base de données, tables) et de la partie physique (devices, fichiers sur disques, ...).

Ce qui différencie donc ASE de la plupart de ses concurrents, et ce qui explique certaines contraintes, c'est que l'on y crée des devices neutres. Ce n'est qu'à l'affection de zones que l'on détermine leur contenu, et que le contenu peut être variable. C'est d'ailleurs pour cela que le stockage des informations sur les devices se fait dans la table *sysdevices* de master, puis que c'est ensuite une table de liens *master..sysusages* qui stocke le partages des devices de *sysdevices* avec les dbid de *sysdatabases*.



sysusages

Microsoft a cassé cette logique dès la version 2000 en rapatriant les informations de stockage dans la base elle-même, et en dissociant fortement les fichiers de données (.mdb) et ceux du journal de transactions (.ldb).

Si l'on se base sur un modèle contraignant, évitant ainsi les écueils, on peut automatiser les choses.

3. Simulation d'autoextension : un exemple

En fait, il suffit de jouer avec la procédure stocké *sp_thresholdaction* qui n'est rien d'autre qu'un configurateur de seuils d'alerte pour les segments.

Basons-nous sur le modèle contraignant suivant:

1. Une base est propriétaire de ses propres fichiers (pas de fichiers partagés entres bases)
2. Le journal de transaction est toujours dissocié
3. On crée arbitrairement 3 types de segments qui auront eux aussi leur(s) propre(s) fichier(s)
4. Chaque segment utilise l'entièreté de son device associé

Il faut avant tout créer une procédure stockée de seuil qui augmentera la taille des devices et des segments au moment critique. Son positionnement dans master est nécessaire à cause de la commande disk remap.

```
use master
go

create procedure dbo.sp_auto_db_etend
( @dbname varchar(30), @segment_name varchar(30),
@space_left int, @status int)
as
begin

declare @v_size varchar(5)
declare @v_device varchar(30)
declare @v_type varchar(5)
declare @v_sql varchar(200)

if @segment_name = 'default'
    select @v_device=@dbname+'_D01',
@v_size='100M', @v_type=''
else
if @segment_name = 'system'
    select @v_device=@dbname+'_D01',
@v_size='100M', @v_type=''
else
if @segment_name = 'indexsegment'
    select @v_device=@dbname+'_I01',
@v_size='100M', @v_type=''
else
if @segment_name = 'logsegment'
    select @v_device=@dbname+'_L01',
@v_size='50M', @v_type='LOG'
else
    begin
    raiserror 20001 'Type de segment non géré'
    return -1
    end

disk resize name = @v_device, size = @v_size

select @v_sql = 'alter database '+@dbname+'
'+@v_type+' ON '+@v_device+'='''+@v_size+''''
print @v_sql
exec (@v_sql)
return 0
end
go
```

La procédure stocké suivante automatise la création d'une base de données selon les critères spécifiques et détermine le seuil critique.

```
user master
go
create procedure sp_auto_db_cree
( @v_dbname varchar ( 30 ))
as
begin

    declare @v_sql varchar ( 200 )
    declare @v_path varchar ( 200 )
    declare @v_file varchar ( 200 )
    declare @v_seg varchar ( 30 )
    declare @v_num_dev int
    declare @v_max_dev int

    /* Contrôle de la préexistance de la base */

    if exists (select name from sysdatabases
where name = @v_dbname)
        begin
        raiserror 20001 "La base  existe déjà"
        return -1
        end


    /* Contrôle du nombre de devices configuré */

    select @v_max_dev=value from sysconfigures
where name = 'number of devices'
    select @v_num_dev=count(*) from sysdevices

    if @v_max_dev < @v_num_dev + 3 -- manque de
structures mémoire pour les devices
        begin
        select @v_num_dev=@v_max_dev + 3
        exec sp_configure 'number of devices',
@v_num_dev
        end

    /* Creation des devices specifiques */

    select @v_path = left ( phyname , datalength
( phyname )- 10 ) from master..sysdevices where
name = 'master'

    select @v_seg=@v_dbname+'_D01',
@v_file=@v_path + @v_dbname + '_DATA01.dat'
    disk init  name= @v_seg, physname=@v_file ,
size='100M'

    select @v_seg=@v_dbname+'_I01',
@v_file=@v_path + @v_dbname + '_INDEX01.dat'
    disk init  name= @v_seg, physname=@v_file ,
size='100M'

    select @v_seg=@v_dbname+'_L01',
@v_file=@v_path + @v_dbname + '_LOG01.dat'
    disk init  name= @v_seg, physname=@v_file ,
size='50M'

    /* Creation de la base */
    select @v_sql = 'create database ' +
```

```
@v_dbname + ' on ' + @v_dbname + '_D01=''100M''
log on ' + @v_dbname + '_L01=''50M'''
    exec ( @v_sql )

    select @v_sql = 'alter database ' + @v_dbname
+ ' on ' + @v_dbname + '_I01=''100M'''
    exec ( @v_sql )

    /* Attribution des segments */
    select @v_sql = @v_dbname+'..sp_addsegment
''indexsegment'', '+@v_dbname+',
'+@v_dbname+'_I01'
    exec (@v_sql)

    select @v_sql = @v_dbname+'..sp_dropsegment
''default'', '+@v_dbname+', '+@v_dbname+'_I01'
    exec (@v_sql)

    select @v_sql = @v_dbname+'..sp_dropsegment
''system'', '+@v_dbname+', '+@v_dbname+'_I01'
    exec (@v_sql)

   /* Délimitation des seuil d'alerte */

   select @v_sql = @v_dbname+'..sp_addthreshold
'+@v_dbname+', system, 5120, sp_auto_db_etend'
    exec (@v_sql)

   select @v_sql = @v_dbname+'..sp_addthreshold
'+@v_dbname+', default, 5120, sp_auto_db_etend'
    exec (@v_sql)

   select @v_sql = @v_dbname+'..sp_addthreshold
'+@v_dbname+', indexsegment, 5120,
sp_auto_db_etend'
    exec (@v_sql)

   select @v_sql = @v_dbname+'..sp_addthreshold
'+@v_dbname+', logsegment, 5120,
sp_auto_db_etend'
    exec (@v_sql)
return 0
end
go
```

Certains choix ont été fait afin de simplifier la démonstration :

- Localisation des devices dans un répertoire unique, à savoir celui hébergeant le device master : en terme de performances, il serait judicieux de choisir des disques distincts pour les divers segments
- Création arbitraire d'un device de 100M (50M pour les logs)
- Augmentation arbitraire d'un device de 100M (50M pour les logs)
- Trois segments distincts (data/system, log et index)

Il sera cependant aisément possible d'améliorer la procédure en "piochant" les informations modifiables via une table de configuration.

Cette solution n'est pas optimum : elle a cependant le mérite de donner les pistes nécessaires à la résolution de cette problématique.

4. Références

1. Diagramme des tables systèmes (Lien79) de Sybase ASE
2. Documentation (Lien79) de Sybase ASE

Retrouvez l'article de fadace : Lien75

Alignement de partitions pour amélioration des performances de MS-SQL Server

Tests et mesures permettant de déterminer l'avantage à aligner les partitions pour un environnement Microsoft SQL Server

1. Introduction

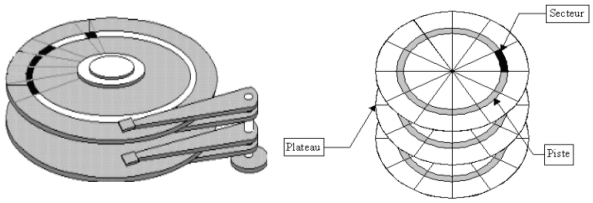
En matière de SGBDR les opérations les plus coûteuses sont celles réalisées par les accès physiques aux disques. Le fractionnement et la structuration des emplacements physiques des disques est d'une importance capitale en terme de performance et peut diviser les temps d'accès aux données par trois. Voici quelques explications sur cette problématique et la façon d' y remédier.

Imaginez que vous êtes bibliothécaire et qu'une partie de votre tâche consiste à ranger des livres en les classant. Aujourd'hui vous devez ranger 2 collections de livres dans une partie de la bibliothèque. Une collection, une fois rangée, remplit la totalité de l'étagère de votre bibliothèque. Vous les rangez donc sur leurs étagères respectives. Si vous devez récupérer une collection entière de livres, il est facile pour vous de le faire : il suffit de repérer l'étagère concernée et de prendre les livres. Maintenant, si votre directeur tient absolument à ranger ses livres au début de cette 1ère étagère les collections de livres qui étaient parfaitement rangés se voient maintenant décalées et dispersées sur 2 étagères à la fois car ceux-ci, rappelons-le, prennent la place d'une étagère entière. Dans ce cas, que ce soit pour les ranger ou les récupérer, il vous faudra évidemment plus de travail.

Remplacez les collections de livres par les données sur vos disques et les livres décalés suite à la demande du directeur par un décalage causé par le système d'exploitation et vous obtiendrez la problématique d'alignement des partitions et des problèmes de performances d'entrées / sorties engendrés.

2. Disque dur : quelques concepts

Commençons par aborder certains concepts techniques concernant les disques durs, sans pour autant entrer dans le détail, l'article ne visant pas l'exhaustivité.



Un disque dur est une mémoire de masse magnétique composée d'un ou plusieurs plateaux. Chaque plateau est composé de pistes. Les pistes situées à un même rayon forment un cylindre. Chaque piste est délimitée en secteurs ou blocs qui contiendront les données. Un secteur fait 512 octets. Un disque dur possède également une ou plusieurs têtes de lecture pour pouvoir lire et écrire les données. Il y a autant de têtes que de surfaces à lire (en fonction des différents plateaux).

3. Partitionnement de disque et 1er secteur

Qu'est ce qu'une partition ? Quelle en est son utilité ?

Une partition accueille un système de fichiers et possède accessoirement un secteur d'amorçage ainsi que la table des partitions (MFT) nécessaire à retrouver vos données. Cette table se situe sur le 1er secteur du disque qui contient également toutes les informations " constructeurs " relatives au disque lui-même. Il s'agit donc du secteur le plus important car il est utilisé par le BIOS pour la reconnaissance du disque.

4. Raid - stripping size et chunck

La plupart des serveurs de bases de données reposent aujourd'hui sur des systèmes à tolérance de panne permettant une haute disponibilité et des performances accrues à base de technologie RAID. Nous terminerons par ce point avant d'aborder l'alignement de partition à proprement parler. La 1ère étape, lorsque configurer un RAID, est de créer votre volume RAID. Vous devrez tôt ou tard choisir une taille pour les " strippings blocks ". Mais qu'est-ce que cela signifie ?

Prenons le cas d'un raid 0. Lors de son fonctionnement, le serveur écrit ou lit sur deux disques en même temps en séparant les données en blocs. Un bloc sur 2 se retrouvera sur un disque et les autres sur le deuxième. Le paramètre " stripping block " permet de régler la taille de ces blocs.

Toutes les tailles conviennent-elles ? Évidemment non ! A moins de posséder une carte professionnelle coûteuse, le découpage des blocs sera assuré par la CPU. Si l'on paramètre une taille trop petite la CPU se trouve surchargée par des opérations de découpage de blocs et le serveur verra ses performances diminuer. Il existe pour chaque contrôleur RAID une taille optimale de bloc appelée " CHUNCK ". Les valeurs de CHUNK courantes varient entre 16Ko et 128Ko. **Pour SQL Server, la taille recommandée est de 64Ko.**

5. Problématique de l'alignement des partitions

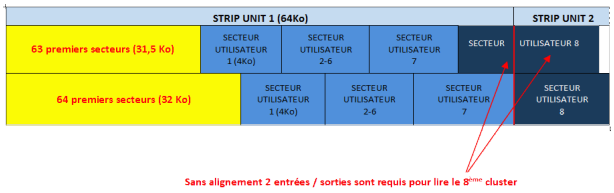
Attardons-nous à présent sur l'alignement des partitions proprement dit. Pourquoi avoir parlé de disque dur, de partitions et de taille de bande (stripe size) ? Lorsque vous créez une partition sur le système d'exploitation, en

l'occurrence Windows (Attention, ceci ne concerne pas Windows Server 2008), par défaut celui-ci se réserve les 63 premiers secteurs du disque dur. Ces 63 premiers secteurs peuvent contenir certaines informations comme le Master Boot Record et la table d'allocations des partitions, comme nous l'avons vu précédemment. Le système d'exploitation écrit ensuite les 512 premiers octets sur la 1ère piste et écrit le reste sur la 2ème piste. Par conséquent une simple écriture ou une simple lecture requiert un double accès ! Nous avons identifié ainsi notre problème, nos livres rangés sur deux étagères….

L'alignement des partitions permet de résoudre ce problème. Comment ? En réservant non pas les 63 mais 64 premiers secteurs du disque. Cette taille peut changer en fonction de la taille d'allocations des clusters de partition. Nous y reviendrons par la suite.

5.1. 1er exemple

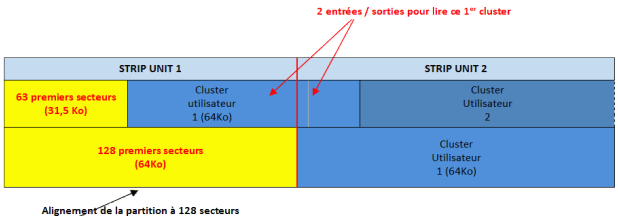
Prenons une valeur de strippe de 64 Ko. Le système d'exploitation alloue une taille de cluster de 4Ko par défaut. Chaque cluster contient donc 8 secteurs de 512 octets et chaque largeur de bande du raid (stripe Unit) contient 16 clusters. L'image ci-dessous illustre nos propos.



Nous voyons bien que pour lire le 8ème cluster il y aura 2 entré

5.2. 2ème exemple

Arrêtons-nous sur un nouvel exemple. Cette fois prenons une taille d'allocation de cluster de 64Ko (recommandée pour les fichiers de données, de transactions et tempdb de SQL Server).



Dans ce 2ème cas nous observons également les avantages d'aligner correctement les partitions en réservant cette fois-ci les 128 premiers secteurs soit 128 x 512 = 64Ko. Comme vous pouvez le constater, la taille d'alignement des partitions se fait en fonction de la taille d'allocation des clusters lorsque vous partitionnez vos disques.

6. Les utilitaires de partitionnement et alignement

De nombreux utilitaires existent pour créer des partitions sur Windows. J'évoquerai simplement quelques utilitaires " natifs " Windows qui nous permettront d'arriver au but recherché, à savoir…l'alignement des partitions !

- **diskmgmt.msc** : la console de management de Windows permet non pas de procéder à l'alignement des partitions mais de faire de l'allocation de cluster.
- **Msinfo32** : cet utilitaire Windows permet de voir l'alignement des partitions (offset) mais ne permet pas de partitionner ou de faire de l'alignement.
- **Diskpar** : cet utilitaire en ligne de commande Windows permet de visualiser et de faire de l'alignement de partitions.
- **Diskpart** : cet autre utilitaire en ligne de commande Windows permet également de faire la même chose que diskpar.

Nous prendrons pour notre exemple " diskpart ".

```
DISKPART > List disk
DISKPART > Select disk 3
DISKPART > create partition primary align=64
DISKPART > assign letter=G
DISKPART >Exit
C:> format /fs:ntfs /A:64K /V:"DiskSql"  /Q   G:
```

Dans l'exemple ci-dessus, nous avons créé une partition primaire sur le disque 3 avec un décalage ou offset de 64Ko. Nous avons ensuite réservé la lettre G et formaté notre partition en NTFS avec une taille de cluster à 64Ko.

Visualisons à présent l'offset créé. Pour cela nous n'utiliserons pas *diskpart* car celui-ci se révèle imprécis à la lecture de l'offset qui s'affiche en Ko. Pour rappel, Windows alloue les 63 premiers secteurs du disque par défaut.

Les calculs suivants permettent de vérifier nos propos:

- Par défaut : 63 * 512 octects = 32256 octets = 31.5 Ko
- Avec alignement : 64 * 512 octets = 32768 octects = 32 Ko

Diskpart dans les 2 cas affichera 32 Ko ! Soyez donc prudent à la lecture de l'offset avec cet utilitaire.

Nous prendrons **wmic**, autre utilitaire fourni par Windows permettant de visualiser l'offset de la partition avec plus de précision :

```
C :> wmic partition get BlockSize,
StartingOffset, Name, Index

BlockSize  Index  Name
StartingOffset
512        0      Disque n° 0, partition n° 0
32256
512        0      Disque n° 1, partition n° 0
32256
512        0      Disque n° 2, partition n° 0
65536
```

Nous voyons donc la partition créée précédemment ainsi que l'offset de 65536 octets soit 64Ko.

Notez au passage le désalignement des disques n°0 et n°1 avec un offset de 32256 octets par défaut soit 31.5Ko et 63 secteurs et non 32Ko et 64 secteurs !

7. Tests et contrôle du gain de performance obtenu

Après avoir aligné et partitionné correctement nos disques, voyons les résultats en termes de performance. Pour cela nous utiliserons **SQLIO**, utilitaire fourni par Microsoft permettant de mesurer la capacité en terme d'entrées / sorties pour une configuration donnée.

Les tests se baseront sur différents scenarii rencontrés dans SQL Server dans un environnement de type OLTP. Nous aurons les caractéristiques suivantes :

- Les lectures des fichiers de données sont constantes par nature et aléatoires (8Ko)
- Les écritures dans les fichiers données se font essentiellement lors des opérations de CHECKPOINT et sont aléatoires. (8Ko)
- Les opérations de type ROLLBACK utilisent un mode de lecture aléatoire sur les fichiers de transactions (8Ko)
- Les lectures des fichiers de transactions sont également séquentiels (La taille des secteurs pouvant aller jusqu'à 120 KB)
- Les écritures dans les fichiers de transactions sont séquentielles et de diverses tailles (tout dépend de la nature de la charge)
- Les opérations de réindexation de tables, d'insertion en blocs et de vérification de bases de données utilisent un mode de lecture et d'écriture séquentiel (64Ko – 256Ko)
- Les backups utilisent un mode d'écriture séquentiel avec une taille de 1024Ko.

Nous testerons pour chaque partition (non alignée et alignée) :

- Pour les I/O de aléatoires : Lecture et écriture avec une taille de bloc à 8Ko
- Pour les I/O séquentielles : Lecture et écriture avec les tailles de bloc de 8, 64, 128, 256 et 1024Ko

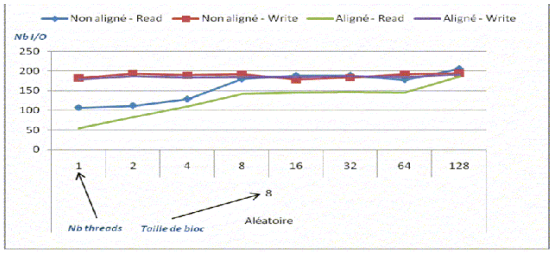
7.1. Environnement du test

Les tests sont réalisés sur un portable ayant les caractéristiques suivantes :

- Intel Core Duo 2,10 Ghz
- 3 Go de RAM
- 2 disques durs partitionnés respectivement avec alignement (offset 64Ko) et sans alignement.
- La taille des blocs est fixée à 64Ko.

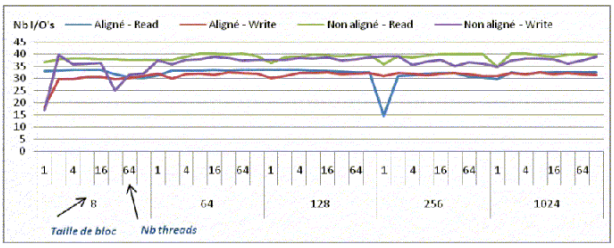
7.2. Résultat des tests

7.2.1. Lectures et écritures aléatoires avec une taille de bloc de 8Ko.



Les tests montrent une différence notable pour lectures (**21%**) mais pas de grande différence en écriture (**1,5%**). Ce test peut être difficilement interprétable. Pourquoi n'observe t'on pas tellement de différence en écriture ? Nous avons une taille de cluster de 8Ko, une largeur de bande de 64Ko et des écritures aléatoires. La probabilité d'écrire dans la zone critique, c'est-à-dire celle qui divisera un bloc en 2 parties reste faible, que ce soit sur une partition alignée ou pas.

7.2.2. Lectures et écritures séquentielles avec une taille de bloc allant de 8ko à 1024Ko



Les tests montrent une différence notable aussi bien en écriture (**13%**) qu'en lecture (**18%**). Sur des lectures et écritures séquentielles la probabilité d'écrire sur la zone critique est beaucoup plus élevée dans ce cas. L'alignement des partitions y joue tout son rôle.

8. Conclusion

L'alignement des partitions reste une des nombreuses techniques d'optimisations existantes telles que la mise en place de technologies RAID, l'ajout de disques rapides, de caches disques, de bus de données plus rapides etc.… Corrélée à un paramétrage adéquat de taille de clusters et de largeur de bande (stripe unit size) aux niveaux de vos configurations RAID, vous devriez tirer parti des avantages d'une technique assez simple à mettre en œuvre avant même d'avoir installé votre serveur SQL.

Bon alignement !!

9. Sources

9.1. Webographie

- Technologies raids wikipédia (Lien81)
- Disque dur wikipédia (Lien81)
- Microsoft Best practices I/O configuration (Lien82)

9.2. Bibliographie

- Outils de partitionnement (Lien83)

Retrouvez l'article de David Barbarin en ligne : Lien84

Compte rendu de l'XP Day Suisse (Genève) 2009

Le 30 Mars 2009 s'est tenue à Genève la conférence XP-Day sur l'eXtreme Programming et les méthodes agiles. Developpez.com y était. En voici le compte-rendu.

1. Présentation

La conférence XP-Day porte sur l'eXtreme Programming et les méthodes agiles. Il existe plusieurs éditions des XP-Day : en France, Belgique, Suisse, etc.

L'édition destinée à la Suisse Romande s'est déroulée le 30 Mars 2009 à Genève, dans le *Geneva Business Center*. Cette édition était entièrement francophone (contrairement à d'autres éditions des XP-Day).

Organisation

La conférence est composée de 12 sessions, qui se déroulent en parallèle sur 2 salles.

La difficulté des sessions est indiquée par un système de ceintures, comme au judo:

- ceinture blanche : débutant
- ceinture jaune : intermédiaire
- ceinture noire : avancé

La salle 1 accueillait les sessions notées « ceinture blanche », tandis que la salle 2 accueillait les sessions d'un niveau plus avancé.

Nous étions 2 pour couvrir la plupart des sessions.

Prix

L'inscription coûte entre 80.- et 100.- CHF, suivant la date d'inscription (80.- CHF si inscrit 6 semaines à l'avance, 100.- CHF ensuite).

Geneva Business Center

Geneva Business Center
Avenue des Morgines, 12
1215 Petit Lancy
Suisse

Accès

L'accès au Geneva Business Center par les transports en commun est très simple car il se situe sur la ligne de bus numéro 23, arrêt « Bossons ».

En revanche, l'accès en voiture n'est pas des plus aisés. La circulation dans Genève est extrêmement difficile, et trouver une place de parking l'est encore plus. Par ailleurs, le parking indiqué comme gratuit sur le site du XP-Day ne l'était malheureusement pas.

Les goodies

À son arrivée, le visiteur se voit remettre un superbe sac en toile de jute griffé « XP day, pour un développement durable ».

On apprécie le parallèle qui est fait entre "développement durable", respect de l'environnement, utilisation de matériaux renouvelables et "développement logiciel" durable et renouvelable.

Les sessions

Pour ce compte-rendu, nous avons décidé de présenter les sessions non pas dans l'ordre chronologique mais en les regroupant par thèmes :

- sessions théoriques
- retours d'expériences
- session philosophique

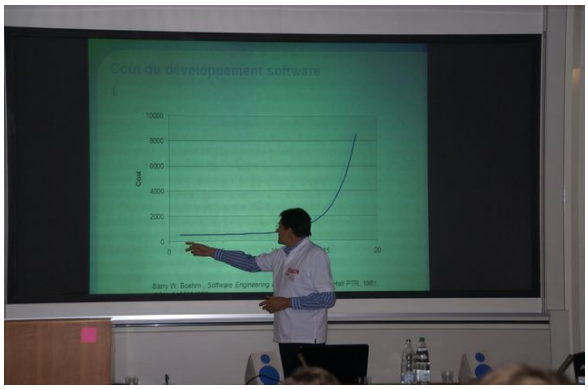
2. Sessions théoriques

Ces sessions ont pour but soit de faire découvrir les méthodes agiles en général, soit de faire découvrir des aspects très particuliers des méthodes agiles.

2.1. Découvrir eXtreme Programming

Cette séance constitue une très bonne présentation d'XP. Destinée principalement aux développeurs n'ayant pas ou peu d'expérience d'XP, cette séance fut accessible, claire et très bien illustrée par de nombreux points et des métaphores.

En tout début de séance, l'intervenant Didier Besset nous explique que le but d'XP est de maîtriser les coûts, en particulier les coûts liés aux changements. En effet, plus des changements doivent survenir tard dans le développement, plus ces changements coûtent cher ; la courbe du coût étant exponentielle :



Pour maîtriser les coûts liés aux changements, XP propose:

- le développement itératif
- une planification adaptable et non rigide
- une architecture évolutive et non contraignante
- la multiplication des tests

Puis il nous explique certains fondamentaux d'XP.

XP c'est:

- faire preuve de bon sens
- faire les choses simplement
- privilégier la communication
- intéragir de près avec le client
- faire tout cela à 100%

Un principe fondamental, c'est de respecter la place du client et des développeurs ainsi que les connaissances de chacun. Ainsi on suppose que:

- le client connaît son métier
- les développeurs connaissent leurs outils
- le client définit les priorités
- les développeurs définissent les détails

Ainsi le client doit / les développeurs ne doivent pas / le client a le dernier mot pour :

- décider du workflow
- définir les champs, menus, boutons
- choisir une charte graphique

Les développeurs doivent / le client ne doit pas / les développeurs ont le dernier mot pour :

- décider d'une architecture
- choisir les outils (langage, BDD)
- définir le modèle

Le client définit les priorités. Si une date de livraison semble en danger, risquant d'être dépassée, les développeurs doivent signaler le problème au client qui décidera des fonctionnalités à retirer afin de conserver la date de livraison.

Après une bonne livraison, c'est le client qui décide de la mise en service.

Après ces explications sur le rôle d'XP, la place du client et des développeurs, Didier Besset explique en détails quelques grands principes d'XP:

Test Driven Design

Les tests unitaires sont écrits par les développeurs. Il faut écrire les tests unitaires avant de coder. Il faut tester tout ce qui peut "foirer" ainsi que le comportement aux valeurs limites.

Les tests d'acceptation sont écrits par le client. Ils permettent de vérifier le respect du cahier des charges (et de savoir quoi tester). Ils sont un bon départ pour la documentation.

Intégration Continue

Chaque changement est immédiatement inclus dans un système déployé. Le déploiement fait partie du développement.

- Un système utilisable existe en permanence:
- les développeurs peuvent essayer le système
- le client aussi

Livraison fréquente

- elle est la conséquence directe de l'intégration continue
- le client a le système en main le plus tôt possible
- le client peut réagir
- le choc des nouvelles fonctionnalités est atténué

Simplicité

Écrire du code lisible :

- les autres doivent pouvoir le lire
- vous aussi, après plusieurs mois

Les intentions doivent être claires :

- utiliser des standards
- éviter les abbréviations
- éviter les commentaires

Le dernier point (éviter les commentaires) peut paraître bizarre mais pour l'intervenant, le code doit se suffire à lui-même. S'il y a des commentaires, c'est que le code n'est pas lisible. J'aurais tendance à dire que les commentaires doivent exprimer ce que le code ne dit pas

(fonctionnement général, invariant, propriétés de certaines variables...) mais Didier Besset n'est pas d'accord sur ce point.

En conclusion, Didier Besset commente cette citation de Peter Merel: *"Make it run, make it run well, make it run fast. In that order."*:

- *"Make it run"* : passer les tests
- *"Make it run well"* : refactoriser, rendre le code lisible
- *"Make it run fast"* : optimiser

Et surtout : ne rien dupliquer !

2.2. Les 5 premiers pas pour devenir agile

La séance est présentée par Portia Tung et Pascal Van Cauvenberghe.

Le duo fonctionne bien et présente sous forme ludique les petits outils/jeux permettant de qualifier/améliorer son agilité. L'objectif de la présentation va au delà des 5 premiers pas pour devenir agile et explore plutôt comment amener une équipe à détecter des problèmes via l'utilisation de jeux.

La partie la plus intéressante vient d'un jeu illustrant la théorie des contraintes. Ce jeu montre comment une personne surchargée peut bloquer très vite une équipe.

Bien sûr, ce jeu est construit pour mettre en évidence ce qu'il veut montrer. L'équipe doit colorier un ensemble de dessins, un des membres se retrouve avec trop de travail pendant que le reste de l'équipe regarde. L'effet 'physique' est saisissant.

Comme tous les jeux très simples, il construit une situation caricaturale. L'important est donc le discours qu'il y a autour. Évidemment tout le monde se doute bien que si une personne se charge de tout, d'autres attendent. Ce qui est plus subtil c'est de se dire que si une personne est surchargée et stressée dans l'équipe, la soulager permet non seulement de lui faciliter la vie mais a de bonnes chances d'améliorer le processus global.

Une personne de l'assistance demande si le concept ne doit pas être poussé jusqu'au bout en 'laissant partir' un leader technique qui prend trop de place dans l'équipe et laisser l'équipe se réorganiser d'elle-même. La conclusion de l'assistance est partagée entre le concept d'équipe homogène et d'équipe 'moyenne'

Lors d'une pause j'ai l'occasion de discuter plus en avant avec Pascal. Il m'explique comment il utilise ces jeux pour faire prendre conscience aux équipes des points d'amélioration possibles. Ainsi un des points clef de son travail de consultant consiste à faire se rencontrer tous les intervenants d'un processus. La tentation pour chaque équipe étant d'optimiser sa partie et non la chaîne globale. Les jeux prennent alors beaucoup d'intérêt pour aller au delà des frottements inévitables entre services.

2.3. TDD et Erlang

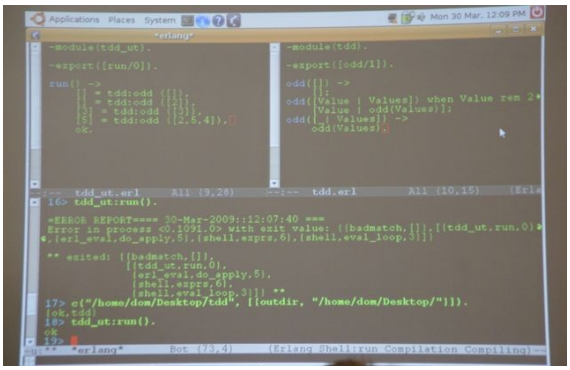
Erlang est un langage de programmation conçu à l'origine par Ericsson avant d'être rendu public. Erlang est à la fois un langage de programmation fonctionnelle et un langage de programmation concurrente.

Cette session, présentée par Dominic Williams et Nicolas Charpentier, a permis de montrer l'application de la TDD à chacun de ces paradigmes de programmation.

Dominic Williams a d'abord montré en quoi la programmation fonctionnelle diffère beaucoup de la programmation procédurale ou Orientée-Objet, y compris en ce qui concerne les tests. En programmation Orientée-Objet, par exemple, avant de pouvoir tester il est nécessaire d'initialiser les objets et d'effectuer une série d'opérations afin d'amener les objets dans l'état que l'on cherche à tester. En programmation fonctionnelle, l'écriture des tests est souvent plus simple, plus lisible (une qualité primordiale en TDD) et semble finalement plus naturelle.

Pour illustrer cela, Dominic Williams a comparé des tests unitaires portant sur des listes : d'abord dans un langage Orientée-Objet, puis en Erlang. Les tests unitaires en Erlang se sont montrés plus concis (une ligne) et plus lisibles car l'initialisation des listes était beaucoup plus simple.

Les démonstrations des tests en Erlang ont été faites en direct, sous Emacs, à la mode « ping-pong » (Dominic écrit les tests, Nicolas modifie le programme en conséquence) :



En haut à gauche : les tests.
En haut à droite : le programme à tester.
En bas : le résultat de la compilation/test.

Une autre démonstration intéressante a permis de montrer comment tester des fonctions d'ordre supérieur.

En programmation fonctionnelle, l'abstraction s'obtient par l'utilisation de fonctions d'ordre supérieur (c'est-à-dire des fonctions acceptant d'autres fonctions comme paramètre). En Programmation Orientée-Objet, l'abstraction se fait par héritage, ce qui pose certains problèmes : un même comportement doit être testé dans les sous-classes, parfois il est même nécessaire de définir des sous-classes dans le seul but de tester le comportement de la super-classe.

Toutes ces démonstrations ont permis de montrer que l'écriture de tests unitaires était simple de par la concision et l'expressivité du langage. Mieux encore : l'approche TDD permet de bien coder en programmation fonctionnelle. Par exemple, la TDD permet de tout de suite

tester (et donc d'implémenter) les cas d'arrêt d'une fonction récursive, ce qui tend à réduire le risque de récursion infinie.

Le deuxième point important de cette présentation concernait la TDD en programmation concurrente et l'intérêt d'Erlang dans ce domaine.

De plus en plus d'applications introduisent des problématiques liées à la concurrence : threads, parallélisation, applications distribuée, ressources partagées...). Il est possible de faire de la programmation concurrente dans de nombreux langages (par exemple en introduisant des threads), le problème est que l'ordre d'exécution est alors non-déterministe et il devient extrêmement difficile d'écrire des tests pour des programmes concurrents.

En Erlang, la notion de programmation concurrente se fait par l'introduction d'« acteurs » indépendants qui communiquent entre eux par l'échange de messages (Erlang gère même les cas de « timeout »). L'exécution devient alors parfaitement déterministe, le programme est simple, facile à comprendre, et il devient facile d'écrire des tests.

Cette présentation fut extrêmement intéressante. Elle a permis de montrer l'intérêt de la TDD en programmation fonctionnelle ainsi que les avantages de Erlang pour résoudre les problèmes de programmation concurrente et faciliter les tests pour ce type de programmes. Cette présentation a également permis de montrer que la programmation fonctionnelle pouvait être utilisée dans certains projets industriels.

Pour bien comprendre cette session, il est utile d'avoir quelques connaissances des bases de la programmation fonctionnelle, pas forcément en Erlang (pour ma part je ne connaissais pas du tout Erlang, j'étais plus habitué à Ocaml).

Toute la difficulté pour les orateurs fut de présenter, en 60 minutes seulement, les différents concepts liés à la programmation fonctionnelle devant un public peu habitué à ce genre de programmation (comme en témoignent les questions posées lors de la séance), et cela avant même de rentrer dans le cœur du sujet : les tests.

Afin que la session reste accessible au plus grand nombre, les exemples choisis restent simples (l'exemple sur les listes est un cas d'école). De fait, les habitués de la programmation impérative s'interrogent sur l'intérêt de la programmation fonctionnelle alors que les habitués de la programmation fonctionnelle (en minorité dans la salle) auraient été friands de cas moins triviaux (ce qui aurait rendu, de fait, la session beaucoup moins accessible).

3. Retours d'expérience

Voici les sessions concernant les retours d'expériences sur XP et les méthodes agiles.

Ces sessions sont intéressantes car elles montrent les applications des méthodes agiles sur des projets réels (parfois d'envergure) et font le point sur les erreurs à éviter.

3.1. XP à dimension industrielle

Une présentation intéressante dans laquelle les intervenants (Pierre-Emmanuel Dautreppe et Norman Deschauwer) font le bilan de l'utilisation d'XP sur un gros projet.

Le contexte du projet est le suivant :

Thales Belgique a été mandaté par une grande banque sociale belge. Le projet était initialement d'un an puis a été renouvelé d'année en année pour finalement s'étendre sur 6 ans. Au départ, le projet regroupait 3 à 4 personnes pour atteindre 15 personnes au final.

Pour ce projet, c'est l'entreprise client qui a imposé l'utilisation d'XP, ce qui fut un gros avantage pour l'équipe en charge du développement. En effet, dans la plupart des projets faisant intervenir XP, l'initiative d'utiliser XP revient souvent à l'équipe informatique qui doit alors convaincre le client du bien-fondé d'XP.

En revanche, l'équipe informatique n'avait pas l'habitude d'utiliser XP et n'avait pas de bonnes pratiques à la base. Ce projet fut donc l'occasion de reparcourir l'ensemble des pratiques qu'offre XP, ainsi que ses valeurs :

- communication
- simplicité
- réactivité
- feedback
- courage (être prêt à repartir de zéro)
- respect (team first)

Il fut donc décidé d'introduire les pratiques d'XP au fur et à mesure, ce qui fut une erreur, d'après les intervenants. Selon eux, il eût mieux valu introduire l'ensemble des pratiques dès le départ.

Intégration continue

L'intégration continue a été mise en place dès le début du projet.

Un serveur dédié compile et lance des tests et envoi des mails lorsque quelque chose ne fonctionne pas. Cela permet d'évaluer rapidement les risques et de communiquer les résultats des tests.

Planning game

Une technique arrivée trop tard dans l'entreprise.

Cette technique permet aux développeurs d'avoir une vision plus globale du projet. Elle permet aux analystes de présenter le besoin fonctionnel du client, sous forme de « user stories ». Ainsi il devient plus facile de définir ce qu'est une tâche.

Communication

Echanger le savoir

Tout le savoir de l'entreprise est centralisé au sein du

service informatique.

L'une des dérives possibles est alors de se sentir meilleur que l'utilisateur et de vouloir dicter son comportement.

Utiliser des métaphores

Les métaphores facilitent la communication. De plus, elles permettent de trouver un langage commun entre les informaticiens et le client. Ainsi les données métiers se retrouvent au niveau du code.

Pour le développement, on parle fonctionnalités plutôt que tâche. Travailler sur une tâche est sans intérêt, travailler sur une fonctionnalité l'est beaucoup plus.

TDD

Une grosse erreur fut de mélanger tests unitaires et tests de recettes. Une autre erreur fut d'écrire du code qui n'était pas couvert par des tests.

Le gros problème, c'est que l'on n'apprend pas à faire des tests à l'école. Écrire un bon test est un exercice compliqué et il est nécessaire de:

- sensibiliser les développeurs à l'écriture de tests
- écrire les tests avant de coder
- écrire des tests lisibles plusieurs mois, voire plusieurs années plus tard

Livraison rapide

Intérêt :

- meilleur R.O.I.
- un feedback rapide
- plus grande réactivité

Les itérations durent 2 semaines, une livraison a lieu tous les 2 mois.

Conception simple

La conception doit être simple, mais simple ne veut pas dire simpliste.

Refactoring

Le refactoring consiste à prendre du recul et à regarder l'application dans son ensemble, puis à passer un grand coup de balais dans l'implémentation.

Le rôle de l'Architecte Agile est de mettre le doigt sur les zones à risque (en terme de performance, de design...) et de procéder à la revue de code.

Avantage : Il connaît l'intégralité du code
Inconvénient : Le fait d'avoir un Architecte Agile peut dé-responsabiliser l'équipe

Retrouvez la suite de l'article de Pierre Caboche et Jean-Philippe Vigniel en ligne : Lien85

Les livres Conception

SOA Security

Anyone seeking to implement SOA Security is forced to dig through a maze of inter-dependent specifications and API docs that assume a lot of prior security knowledge on the part of readers. Getting started on a project is proving to be a huge challenge to practitioners. This book seeks to change that. It provides a bottom-up understanding of security techniques appropriate for use in SOA without assuming any prior familiarity with security topics.

Unlike most other books about SOA that merely describe the standards, this book helps readers learn through action, by walking them through sample code that illustrates how real life problems can be solved using the techniques and best practices described in the standards. It simplifies things: where standards usually discuss many possible variations of each security technique, this book focuses on the 20% of variations that are used 80% of the time. This keeps the material covered useful for all readers except the most advanced.

This book shows you

- Why SOA Security is different from ordinary computer security, with real life examples from popular domains such as finance, logistics, and Government
- How things work with open source tools and code examples as well as proprietary tools.
- How to implement and architect security in enterprises that use SOA. Covers WS-Security, XML Encryption, XML Signatures, and SAML.

Critique du livre par Gildas Cuisinier

SOA et Sécurité, deux termes très présents et très importants dans le développement en entreprise.
Le titre est donc très prometteur. Mais les auteurs réduisent tout de suite la portée du livre dans l'introduction.

En effet, ce livre n'explique pas l'architecture et les concepts SOA et n'expose pas toutes les notions de sécurité non plus. Le livre portant sur l'intersection de ces deux domaines, une connaissance minimale des deux sujets était nécessaire pour commencer, ce livre s'adresse donc à un public initié, mais sans pour autant expert.
Cela dit, le livre est tout de même bien ficelé et intéressant. Il est composé de trois parties.

La première consiste en un rappel des bases de la SOA et de la sécurité par WebService : SOAP, SOAP Header, WS-Security.

La seconde partie présente les concepts de sécurité : authentification, autorisation, chiffrement,… Cette section est particulièrement intéressante. Elle présente diverses pratiques (utilisateur/mot de passe, Kerberos, PKI) tout en décrivant leurs avantages et inconvénients.

La dernière partie est légèrement plus complexe et traite réellement de la sécurité orientée service. Encore une fois, les différentes implémentations d'un service de sécurité sont présentées ainsi que les technologies utilisées dans ce but (SAML, WS-Trust, …)

A la fin de la lecture, on a acquis bon nombre d'informations, mais il subsiste toutefois une impression d'insuffisance sur le sujet. Mais encore une fois, c'est volontaire. Au vu de la complexité des sujets, seules les bases sont présentées mais une multitude de liens sont fournis pour ceux qui désirent approfondir un sujet précis.

Au niveau des exemples, une implémentation basée sur Axis est fourni en fin de chapitre. C'est sans doute le seul bémol, Axis étant un peu vieillissant. Cependant, les exemples sont suffisamment explicites pour être adaptables facilement avec n'importe quel autre framework.

Ce livre est donc plus qu'intéressant, même si le titre " Introduction to SOA Security " aurait été plus représentatif.

Management d'un projet système d'information

Cet ouvrage s'adresse aux responsables de systèmes d'information et aux chefs de projets, ainsi qu'aux étudiants en informatique ou système d'information et aux élèves ingénieurs. Quelle est la meilleure façon de conduire un projet système d'information ? Ce livre répond à cette interrogation en analysant les outils et les méthodes de gestion du domaine à partir des points clés que sont : l'analyse et le découpage d'un projet ; l'évaluation des risques ; l'estimation des charges ; les techniques de planification ; l'organisation du travail ; la dimension humaine et relationnelle du projet ; le pilotage du projet; la maîtrise et la qualité du projet. les principales normalisations internationales. Chacun de ces points clés fait l'objet d'exemples de mise en œuvre, d'exercices et d'études de cas détaillés et explicités. La planification et le pilotage d'un projet sont illustrés avec le progiciel MS Project 2003. De plus, l'ouvrage apporte une aide à la préparation de la certification en management de projet du PMI. Cette sixième édition introduit pour chaque aspect du management de projet une perspective particulière sur les méthodes agiles.

Critique du livre par Matthieu Brucher

D'habitude, un livre sur les méthodes de développement d'un système d'information est assez basique : on parle du cycle en V, on fait le tour du propriétaire et voilà.

Ici, aucun parti n'est pris. Les cycles en V côtoient SCRUM et XP, ce qui est rassurant. Avant les exemples, les différentes étapes de la gestion de projet sont envisagées. Estimation des charges, planification, gestion de l'équipe, pilotage et surtout qualité.

Les exemples sont organisés comme des exercices basés sur des cas réels. L'intérêt par rapport à d'autres ouvrages est que ces exemples prennent la majeure partie de celui-ci. On a donc une bonne description de la théorie, mais la pratique n'est pas le maillon faible. Et surtout, on a un corrigé "type" (même si les avis peuvent toujours légèrement diverger), avec argumentation, ...

Donc ce livre est un vrai ouvrage de référence (à mon avis) pour quelqu'un qui débute. Il ne devrait pas avoir de difficulté à utiliser les différents exercices et exemples pour gérer son propre projet.

Retrouvez ces critiques de livre sur la page livres Conception : Lien86

Integrer Ogre à Qt - Partie 2

Ce tutoriel fait suite à "Intégrer Ogre à Qt" (Lien87). A la fin de ce dernier, nous avons obtenu un embryon d'outil, permettant assez peu de choses tel quel, mais ayant fourni les bases principales requises. Faisant suite à ce tutoriel, je vais présenter ici 2 améliorations. La première consiste à simplifier le travail à fournir par le développeur quant aux actions à implémenter, tandis que la seconde se penche sur une amélioration critique pour tout utilisateur: la possibilité d'annuler ses actions.
Nous repartons ici du code final du tutoriel précédent dont voici le lien (Lien88).

1. Une autre gestion des événements

1.1. Inconvénients de l'approche précédente et motif pour changer

A la fin de l'article précédent, nous pouvions déplacer la caméra par 3 méthodes différentes. Cependant, l'approche qui a été choisie n'est pas viable à long terme dû à une conception trop monolithique. En effet, simplement ajouter la possibilité de déplacer l'entité sélectionnée demanderait de complexifier les implémentations d'événements existants. Le résultat serait des fonctions particulièrement "bloatée", et au mieux un enchaînement de if ou un switch pour basculer entre divers gestionnaires d'événements selon le but à atteindre.
Nous allons changer cette approche pour une méthode plus souple. Cette méthode mettra en oeuvre le design pattern Strategie afin de découpler au maximum les réponses aux événements du widget en lui même. Cette technique va nous permettre de déléguer la gestion des événements du widget. Chaque manipulation du widget se verra isolée dans sa propre classe, et les interactions possibles en hériteront. Du point de vue de la classe OgreWidget, cela nécessite de garder à jour un pointeur sur la classe de base afin de lui transférer les événements (si délégué actif il y a; nous nous laissons la possibilité de ne pas avoir de gestion d'événements en cas de pointeur nul).
Pour plus d'infos sur le pattern Stratégie ainsi que des exemples en C++, je vous recommande de lire cette section d'un tutoriel de David Come (Lien89).

1.2. Implémentation

1.2.1. Définition de l'interface nécessaire et modification au code actuel

Nous allons donc définir une classe indiquant toutes les fonctions requises à un gestionnaire d'entrées. Il s'agira dans notre cas des méthodes interagissant sur la scène, à l'exception du double clic (ce dernier est "réservé" à la (dé)sélection d'entité). La classe en question est déclarée ainsi:

```
class EventHandler
{
public:
    virtual bool keyPressEvent(QKeyEvent *e);
    virtual bool mouseMoveEvent(QMouseEvent *e);
    virtual bool mousePressEvent(QMouseEvent *e);
    virtual bool mouseReleaseEvent(QMouseEvent
*e);
    virtual bool wheelEvent(QWheelEvent *e);
};
```

Ces méthodes ne sont pas sensées spécifier si l'événement est accepté ou non. Cette réponse est laissée au soin de notre widget qui indiquera l'acceptation de l'événement selon que la fonction renvoie **true** ou **false**. Vous l'aurez peut-être remarqué, EventHandler n'est pas une classe abstraite, mais une classe no-op. En effet, chacune des méthodes est implémentée ici pour retourner false. Ca permet de ne réimplémenter qu'une partie des fonctions dans les classes dérivées. Je me passerais donc ici de montrer l'implémentation (une série de return false n'étant pas des plus intéressante ;)).

Il va nous falloir maintenant intégrer ce type à **OgreWidget**. Pour ce faire, il faut commencer par ajouter une variable à cette classe sous la forme d'un pointeur; ceci nous permettra de changer le gestionnaire d'événements selon les besoins. Le corps des méthodes gérant les événements listés dans EventHandler est donc tous remplacé par un code similaire à celui-ci (aucun intérêt à tous les lister):

```
void OgreWidget::keyPressEvent(QKeyEvent *e)
{
    if(eventHandler && eventHandler-
>keyPressEvent(e))
    {
        e->accept();
    }
    else
    {
        e->ignore();
    }
}
```

La vérification sur l'existence d'un eventHandler (en supposant que vous assigniez bien vos pointeurs à 0 une fois supprimés) permet de facilement assigner à OgreWidget un EventHandler (ou classe dérivée) qui ne fait rien, sans pour autant assigner un EventHandler. Il nous faut aussi une méthode pour changer dynamiquement ce comportement, et c'est réalisé par la fonction

OgreWidget::setEventHandler:

```
void OgreWidget::setEventHandler(EventHandler
*newEventHandler)
{
    delete eventHandler;
    eventHandler = newEventHandler;
}
```

1.2.2. Implémentation du déplacement de la caméra

Après avoir apporté les modifications ci-dessus, nous ne pouvons plus que sélectionner et désélectionner le robot. Le but de cette section est donc de rétablir le comportement précédent. Cependant, en ayant délocalisé la gestion des entrées, nous n'avons plus la main sur l'ensemble des chemins possibles déclenchant une modification de la caméra. Il en résulte une impossibilité de garder la position synchrone avec les spinbox. Nous allons donc découpler un peu tout ça, et commencer par créer un wrapper pour la caméra. Nous n'allons redéfinir que les fonctions que nous allons utiliser ici, et en profiter pour implémenter des slots et signaux:

```
class MyCamera : public QObject
{
Q_OBJECT

public:
    MyCamera(Ogre::Camera &camera, QObject
*parent = 0);

    Ogre::Camera* getOgreCamera() const;
    const Ogre::Vector3& getPosition() const;
    Ogre::Ray getCameraToViewportRay(Ogre::Real
screenx, Ogre::Real screeny) const;

public slots:
    void setPosition(const Ogre::Vector3 &pos);
    void lookAt(const Ogre::Vector3
&targetPoint);
    void setAspectRatio(Ogre::Real ratio);

signals:
    void positionChanged(const Ogre::Vector3
&pos);
    void visiblePropertyChanged();

private:
    Ogre::Camera *ogreCamera;
};
```

Quelques explications sur cette classe restent à faire. Le constructeur prend une référence afin de rendre impossible le passage d'un pointeur null. Les setters sont implémentés comme des slots afin de faciliter le réglage par des widgets externes afin de ne pas encombrer la classe principale (ici OgreWidget) de code qui ne lui est pas spécifique. Le signal visiblePropertyChanged() est là pour signaler qu'un changement a eu lieu sur une propriété qui modifie potentiellement le rendu (changement de position, d'orientation, etc.. etc...). Les getters ne sont pas vraiment intéressants, ils se contentent de transmettre l'appel à la caméra Ogre. L'implémentation des setters par contre se présente ainsi:

```
void MyCamera::setPosition(const Ogre::Vector3
&pos)
{
    ogreCamera->setPosition(pos);
    lookAt(Ogre::Vector3(0,50,0));

    emit positionChanged(pos);
    emit visiblePropertyChanged();
}

void MyCamera::lookAt(const Ogre::Vector3
&targetPoint)
{
    ogreCamera->lookAt(targetPoint);
    emit visiblePropertyChanged();
}

void MyCamera::setAspectRatio(Ogre::Real ratio)
{
    ogreCamera->setAspectRatio(ratio);
    emit visiblePropertyChanged();
}
```

Après ces modifications, il va falloir nettoyer un peu la classe OgreWidget afin de ne pas laisser de code inutile. Les modifications consistent à:

- Remplacer le pointeur Ogre::Camera* par un MyCamera*
- Ajouter une méthode permettant d'obtenir le pointeur vers MyCamera associé à la vue
- Supprimer le slot *setCameraPosition* ainsi que le signal *cameraPositionChanged*
- Supprimer les variables liées à la gestion du déplacement de la caméra (oldPos, turboModifier et invalidMousePoint)
- Et ajouter une méthode *setupCamera()* afin de séparer un peu la création de la caméra de l'initialisation dans un souci de propreté. Voici le code de cette méthode:

```
void OgreWidget::setupCamera()
{
    delete camera;
    camera = new MyCamera(*ogreSceneManager-
>createCamera("myCamera"), this);

    connect(camera,
SIGNAL(visiblePropertyChanged()), this,
SLOT(update()));

    camera->setPosition(Ogre::Vector3(0,
50,150));
    camera->lookAt(Ogre::Vector3(0,50,0));
    camera->setAspectRatio(Ogre::Real(width()) /
Ogre::Real(height()));
}
```

Et la déclaration actuelle de OgreWidget est donc:

```
class OgreWidget : public QWidget
{
[...]
    MyCamera *getCamera();
    void setEventHandler(EventHandler
*newEventHandler);

[...]

private:
[...]
```

```
    void setupCamera();

private:
    Ogre::Root          *ogreRoot;
    Ogre::SceneManager  *ogreSceneManager;
    Ogre::RenderWindow  *ogreRenderWindow;
    Ogre::Viewport      *ogreViewport;

    EventHandler                *eventHandler;
    MyCamera            *camera;

    Ogre::SceneNode     *selectedNode;
};
```

Il nous reste maintenant 2 étapes à franchir avant de pouvoir à nouveau déplacer la caméra: créer un event handler permettant d'agir sur la caméra, et modifier la classe *MainWindow* afin de prendre tout ceci en compte.
Nous allons commencer par créer la classe *CameraEventHandler*, qui va permettre de déplacer la caméra par les méthodes déjà présentées. Cette classe reprend tout simplement l'ancien corps des méthodes d'événements, à l'exception des event->accept() et event->ignore(). Je ne montre ici que la déclaration puisque l'implémentation est déjà connue. En cas de doute, n'hésitez pas à vous référer à l'archive indiquée en fin de paragraphe.

```
class CameraEventHandler : public EventHandler
{
public:
    CameraEventHandler(MyCamera *targetCamera);

    virtual bool keyPressEvent(QKeyEvent *e);
    virtual bool mouseMoveEvent(QMouseEvent *e);
    virtual bool mousePressEvent(QMouseEvent *e);
    virtual bool mouseReleaseEvent(QMouseEvent
*e);
    virtual bool wheelEvent(QWheelEvent *e);

private:
    static const Ogre::Real turboModifier;
    static const QPoint invalidMousePoint;

private:
    QPoint oldPos;
    MyCamera *camera;
};
```

Nous allons finir avec les modifications à apporter à la classe MainWindow. Nous allons activer/désactiver le déplacement de la caméra par le biais d'un menu. Il nous faut donc créer une nouvelle action et l'ajouter au menu Divers. Lorsque l'état de cette action est basculé, ce sera le (nouveau) slot *moveCamModeToggled(bool)* qui sera exécuté. Selon l'état de l'action nous allons activer ou désactiver le déplacement de la caméra, afficher ou masquer le dock contenant notre widget modifiant des coordonnées et connecter les changements de coordonnées entre la caméra et le widget Coordinate3DModifier. Voici les méthodes qui ont changées:

```
    MainWindow()
    :ogreWidget(0)
    {
        ogreWidget = new OgreWidget;
        camPosModifier = new
Coordinate3DModifier;

        createActionMenus();
        createDockWidget();

        setCentralWidget(ogreWidget);
    }

[...]

    void createActionMenus()
    {
        QAction *changeColorAct = new
QAction("Changer la couleur de fond", this);
        connect(changeColorAct,
SIGNAL(triggered()), this,
SLOT(chooseBgColor()));

        QAction *moveCamModeAct = new
QAction("Deplacement de la camera", this);
        moveCamModeAct->setCheckable(true);
        moveCamModeAct->setChecked(false);
        connect(moveCamModeAct,
SIGNAL(toggled(bool)), this,
SLOT(moveCamModeToggled(bool)));

        QAction *closeAct = new
QAction("Quitter", this);
        connect(closeAct, SIGNAL(triggered()),
this, SLOT(close()));

        QMenu *menu = menuBar()-
>addMenu("Divers");
        menu->addAction(changeColorAct);
        menu->addAction(moveCamModeAct);
        menu->addAction(closeAct);
    }

[...]


private slots:
[...]

    void moveCamModeToggled(bool on)
    {
        if(on)
        {
            MyCamera *cam = ogreWidget-
>getCamera();
            CameraEventHandler *camEventHandler =
new CameraEventHandler(cam);
            ogreWidget-
>setEventHandler(camEventHandler);

            coordModifier->setNewCoordinate(cam-
>getPosition());
            coordModifierDock->setVisible(true);

            connect(coordModifier,
SIGNAL(coordinateChanged(const Ogre::Vector3&)),
                    cam, SLOT(setPosition(const
Ogre::Vector3&)));
            connect(cam,
SIGNAL(positionChanged(const Ogre::Vector3&)),
                    coordModifier,
SLOT(setNewCoordinate(const Ogre::Vector3&)));
        }
        else
        {
            ogreWidget->setEventHandler(0);
```

```
            coordModifierDock->setVisible(false);
            coordModifier->disconnect();
        }
    }
```

Une archive contenant tout le code nécessaire à cette partie est disponible (Lien90).

1.2.3. Implémentation du déplacement de l'objet sélectionné

Le processus à suivre suit de très près celui utilisé pour la caméra ; je plongerais donc un peu moins dans les détails.
A l'image de la caméra pour laquelle nous avons créé un wrapper, nous allons en créer un pour la classe Ogre::SceneNode. Il s'agit simplement d'une version "allégée" du wrapper de la caméra dont voici la déclaration:

```
class MySceneNode : public QObject
{
Q_OBJECT

public:
    MySceneNode(Ogre::SceneNode &node, QObject
*parent = 0);

    Ogre::SceneNode* getOgreSceneNode() const;
    const Ogre::Vector3& getPosition() const;

public slots:
    void setPosition(const Ogre::Vector3 &pos);

signals:
    void positionChanged(const Ogre::Vector3
&pos);

private:
    Ogre::SceneNode *ogreSceneNode;
};
```

L'implémentation de cette classe est particulièrement triviale (il s'agit simplement de transférer les appels au SceneNode wrappé).
C'est cette classe qui va être utilisée pour stocker l'objet sélectionné. Nous allons commencer par mettre à jour le code de OgreWidget::mouseDoubleClickEvent() afin d'utiliser cette nouvelle classe. Voici le snippet concerné:

```
if(queryResultIterator != queryResult.end())
{
  if(queryResultIterator->movable)
  {
    Ogre::SceneNode *node = queryResultIterator-
>movable->getParentSceneNode();
    node->showBoundingBox(true);
    delete selectedNode;
    selectedNode = new MySceneNode(*node, this);
  }
}
else
{
  if (selectedNode)
  {
    selectedNode->getOgreSceneNode()-
>showBoundingBox(false);
    delete selectedNode;
    selectedNode = 0;
  }
}
```

Maintenant, il nous faut une dernière modification à OgreWidget afin de permettre à l'interface de déplacer l'objet. Nous ajoutons une méthode (OgreWidget::getSelectedSceneNode()) nous permettant d'obtenir l'objet sélectionné.
Passons maintenant à l'event handler à qui l'on délègue la gestion des événements pour déplacer le noeud concerné. Le comportement est semblable à l'event handler, à 3 exceptions près:

- il n'y a pas de modificateur "turbo", et le déplacement se fait par incrément d'une unité,
- le déplacement à la souris se fait sur le plan XZ, contrairement à la caméra où la translation se fait en XY,
- les événements de la molette sont ignorés (SelectedNodeEventHandler::wheelEvent() renvoie false).

Il ne nous reste donc plus qu'à créer une entrée dans le menu afin de connecter tout ça. Cette action affichera le widget Coordinate3DModifier, mais cette fois, il affichera et permettra de mettre à jour la position du noeud sélectionné. Cette entrée de menu est mutuellement exclusive avec celle permettant de déplacer la caméra.
Pour implémenter cette exclusivité, nous allons utiliser un QActionGroup (Lien91). Cette classe nous simplifiera la vie en nous permettant de ne pas nous soucier de désactiver l'event handler que nous aurions pu activer précédemment, ni de décocher l'entrée du menu correspondante. Voici le code qui fait ceci:

```
    QActionGroup *actionGroup = new
QActionGroup(this);
    actionGroup->addAction(moveCamModeAct);
    actionGroup->addAction(moveSelNodeModeAct);
```

Oui oui, c'est tout :)
moveSelNodeModeAct est une action qui est créée de la même façon que *moveCamNodeAct*. Seul son texte ainsi que son slot change. Slot dont voici d'ailleurs le code:

```
void MainWindow::moveSelectedNodeModeToggled(bool
on)
{
    if(on)
    {
        MySceneNode *selNode = ogreWidget-
>getSelectedSceneNode();
        if (!selNode)
        {
            moveSelNodeModeAct-
>setChecked(false);

            // On récurse afin de nettoyer
correctement les signaux/slots et event handlers
            moveSelectedNodeModeToggled(false);
            return;
        }
        SelectedNodeEventHandler
*selNodeEventHandler = new
SelectedNodeEventHandler(selNode);
        ogreWidget-
>setEventHandler(selNodeEventHandler);
```

```
        coordModifier->disconnect();
        coordModifier->setNewCoordinate(selNode-
>getPosition());
        coordModifierDock->setVisible(true);
        coordModifierDock-
>setWindowTitle("Selected node position");

        connect(coordModifier,
SIGNAL(coordinateChanged(const Ogre::Vector3&)),
                selNode, SLOT(setPosition(const
Ogre::Vector3&)));
        connect(selNode,
SIGNAL(positionChanged(const Ogre::Vector3&)),
                coordModifier,
SLOT(setNewCoordinate(const Ogre::Vector3&)));
        connect(selNode,
SIGNAL(positionChanged(const Ogre::Vector3&)),
                ogreWidget, SLOT(update()));
    }
    else
    {
        ogreWidget->setEventHandler(0);

        coordModifierDock->setVisible(false);
        coordModifier->disconnect();
    }
}
```

Nous commençons bien sûr par nous assurer qu'un objet est bel et bien sélectionné; dans le cas contraire nous décochons le menu pour signifier que la demande n'est pas acceptée. En l'état, vous pouvez commencer à tester, mais il nous reste un dernier problème à résoudre.
En effet, si vous sélectionnez un objet, que vous entrez en mode de déplacement d'objet, et que vous le désélectionnez, tout événement que nous traitons entraînera un crash puisque le sceneNode piloté est détruit par OgreWidget. Nous allons donc ajouter un signal à cette classe afin de notifier d'un changement de sélection. Voici le signal ajouté:

```
signals:
    void selectionChanged(MySceneNode
*newSelectedNode);
```

Nous passons en paramètre le nouveau noeud par commodité bien qu'il n'y en ait pas d'utilité dans le cas présent. Il nous faut maintenant écrire un slot dans MainWindow afin de mettre à jour l'interface selon le besoin:

```
void MainWindow::selectionChanged(MySceneNode * /
*newSelectedNode*/)
{
    if (moveSelNodeModeAct->isChecked())
        moveSelectedNodeModeToggled(true);
}
```

Ce slot se contente de mettre à jour l'event handler selon la sélection si nous sommes dans le mode de déplacement d'objet. Maintenant, nous émettons ce signal à tout changement de sélection (une petite modification de OgreWidget::mouseDoubleClickEvent() est nécessaire). Enfin, après avoir connecté le signal fourni par OgreWidget à MainWindow::selectionChanged(MySceneNode *), il ne nous reste plus qu'à compiler et exécuter notre petite application afin de constater que nous pouvons tour à tour modifier la position de la caméra et de l'objet sélectionné.

Le code illustrant ce paragraphe est disponible (Lien92).

2. Support de l'annulation

2.1. Présentation du framework Undo fourni par Qt

La possibilité d'annuler et réappliquer des modifications dans un document est devenue extrêmement courant. C'est en effet une fonctionnalité essentielle pour faciliter l'utilisation de votre logiciel. Les éditeurs 3D ou encore de niveaux n'échappent pas à la règle, nous allons donc voir comment implémenter ce système avec Qt.
Le framework Undo de Qt est basé sur le design pattern "Command". Je vous encourage à lire la définition écrite par Baptiste Wicht sur ce même site (Lien93). Pour une description plus détaillée avec un exemple complet, vous pouvez vous référer à cette page (Lien94).
L'idée est donc de wrapper nos actions sur la scène dans des objets, des instances d'une classe dérivant de QUndoCommand (Lien95). Lors de toute modification de la scène, un tel objet sera donc créé de façon à définir comment annuler la modification, ainsi que la façon de l'appliquer. Chacun de ces objets sera poussé dans une QUndoStack (Lien96). Cette dernière est en générale propre à chaque document. La dernière classe que nous utiliserons ici sera QUndoView (Lien97). Elle permet d'afficher le contenu d'une QUndoStack, ainsi que d'y naviguer (c'est-à-dire cliquer sur un état pour voir le document tel qu'il l'était lors de ce snapshot).
Dans le cas d'une application permettant de gérer plusieurs documents, l'utilisation de QUndoGroup (Lien98) est recommandée. L'expression "plusieurs documents" est à prendre au sens général du terme. Prenons un éditeur 3D: vous pouvez vouloir gérer une pile d'undo pour les modifications de la scène, ainsi qu'une autre indépendante pour gérer un panel de matériau. Cette classe ne sera pas utilisée ici.

2.2. Implémentation

2.2.1. Préparatif

Avant d'entrer dans les détails de chaque modification, nous allons mettre en place le nécessaire pour supporter l'undo/redo.
Voici les éléments que nous voulons présenter:

- ajout au menu "Divers" d'une action "Annuler" suivi d'une description de l'action,
- ajout au menu "Divers" d'une action "Refaire" suivi d'une description de l'action,
- annulation du changement de couleur de fond ainsi que du déplacement d'une entité,
- possibilité de fusionner des déplacements de l'entité si le temps écoulé entre 2 commandes est de moins de 5 dixièmes de seconde (afin de prendre en compte le déplacement constant à la souris, au clavier ou aux spinners) et que l'entité est la même que la précédente,
- possibilité d'afficher la pile des modifications.

La première étape consiste à créer une QUndoStack. Notre petit "éditeur" étant mono-document, nous allons la gérer à partir de la classe MainWindow. Une fois doté d'une instance de cette classe, nous allons ajouter les entrées de menus permettant de faire et défaire des actions:

```
QAction *undoAct =
undoStack.createUndoAction(this, "Annuler : ");
undoAct->setShortcut(QKeySequence("Ctrl+Z"));

QAction *redoAct =
undoStack.createRedoAction(this, "Refaire : ");
redoAct-
>setShortcut(QKeySequence("Ctrl+Shift+Z"));
```

Ces actions seront automatiquement mises à jour par QUndoStack lors de l'insertion ou du retrait d'éléments.
Nous allons maintenant ajouter "l'inspecteur", ce petit widget permettra d'avoir une vue sur l'évolution de la pile d'annulation. Il sera présenté dans un QDockWidget dont l'affichage sera pilotable par une entrée dans le menu "Divers". Voici le code nécessaire à la création de ce dock:

```
// Constructeur de MainWindow:
    undoView = new QUndoView(&undoStack, this);
    undoView-
>setSizePolicy(QSizePolicy::Expanding,
QSizePolicy::Expanding);
    undoView->setEmptyLabel("<Scene initiale>");
    undoView->setMinimumWidth(150);
    undoView->setMaximumWidth(150);

// [...]

// Ajout à MainWindow::createActionMenus():
QAction *undoHistoryAct = undoViewDock-
>toggleViewAction();
menu->addAction(undoHistoryAct);

// [...]
```

```
// Ajout à MainWindow::createDockWidget():
    undoViewDock = new QDockWidget(this);
    undoViewDock-
>setAllowedAreas(Qt::LeftDockWidgetArea |
Qt::RightDockWidgetArea);
    undoViewDock-
>setFeatures(QDockWidget::DockWidgetMovable |
QDockWidget::DockWidgetFloatable |
QDockWidget::DockWidgetClosable);
    undoViewDock->setWidget(undoView);
    undoViewDock->setWindowTitle("Historique");
    undoViewDock->setVisible(true);
    addDockWidget(Qt::LeftDockWidgetArea,
undoViewDock);
```

Prenez bien garde à appeler createDockWidget **avant** createActionMenus si vous vous servez du QAction fourni par QDockWidget pour piloter son affichage.
Il nous reste une dernière chose à faire avant de passer à l'implémentation des commandes que nous avons définies précédemment. Afin de pouvoir fusionner des commandes, nous allons avoir besoin d'un identifiant. Pour ce faire, j'ai choisi ici de les (enfin, "le" dans le cadre de ce tuto) stocker dans un en-tête sous la forme d'un enum:

```
#ifndef COMMANDS_H
#define COMMANDS_H

enum CommandID
{
    CID_MoveEntity
};

#endif
```

Et nous sommes enfin prêt à implémenter nos commandes !

Retrouvez la suite de l'article de Denys Bulant en ligne : Lien99

Liens

Lien1 : http://java.developpez.com/faq/javafx/
Lien2 : http://javasearch.developpez.com/
Lien3 : http://www.developpez.com/hebergement/
Lien4 : http://www.toursjug.org/
Lien5 : http://www.lyonjug.org/
Lien6 : http://www.developpez.net/forums/d717308/java/communaute-java/creation-application-type-crud-jsf-jpa/
Lien7 : http://demo.java.developpez.com/crud-jsf-jpa/
Lien8 : http://www.developpez.net/forums/u9119/lunatix/
Lien9 : http://java.developpez.com/equipe/
Lien10 : http://www.developpez.net/forums/d717520/java/communaute-java/rubrique-java-heberge-desormais-sites-web-java/
Lien11 : http://www.gdawj.com/
Lien12 : http://java.developpez.com/livres/
Lien13 : http://lmellouk.developpez.com/tutoriels/jsf/richfaces/
Lien14 : http://mikael-robert.developpez.com/tutoriels/java/seam/composant/facelet/
Lien15 : http://g-ernaelsten.developpez.com/tutoriels/excelphp/
Lien16 : http://www.codeplex.com/PHPExcel/
Lien17 : http://pear.php.net/package/Spreadsheet_Excel_Writer
Lien18 : http://fpdf.org/
Lien19 : http://www.codeplex.com/PHPExcel/Release/ProjectReleases.aspx
Lien20 : http://snaps.php.net/
Lien21 : http://g-ernaelsten.developpez.com/tutoriels/excel2007/
Lien22 : http://www.alistapart.com/articles/understandingprogressiveenhancement
Lien23 : http://en.wikipedia.org/wiki/Graceful_degradation
Lien24 : http://www.hesketh.com/publications/inclusive_web_design_for_the_future/
Lien25 : http://en.wikipedia.org/wiki/Progressive_enhancement
Lien26 : http://developer.yahoo.com/yui/articles/gbs/
Lien27 : http://www.flickr.com/photos/aarongustafson/83123599/
Lien28 : http://www.stuffandnonsense.co.uk/archives/web_standards_trifle.html
Lien29 : http://en.wikipedia.org/wiki/Chocolate-coated_peanut
Lien30 : http://s-jdm.developpez.com/tutoriels/web/traductions/amelioration-progressive-css/
Lien31 : http://s-jdm.developpez.com/tutoriels/web/traductions/amelioration-progressive-javascript/
Lien32 : http://s-jdm.developpez.com/tutoriels/web/traductions/comprendre-amelioration-progressive/
Lien33 : http://www.jankoatwarpspeed.com/post/2008/05/22/CSS-Message-Boxes-for-different-message-types.aspx
Lien34 : http://itweek.deviantart.com/art/Knob-Buttons-Toolbar-icons-73463960
Lien35 : http://dmouronval.developpez.com/tutoriels/css/messages-personnalises/fichiers/exemple.html
Lien36 : http://dmouronval.developpez.com/tutoriels/javascript/jquery/animer-messages/
Lien37 : http://dmouronval.developpez.com/tutoriels/css/messages-personnalises/
Lien38 : http://falola.developpez.com/tutoriels/javascript/namespace/
Lien39 : http://cmanager.servebox.org/
Lien40 : http://maven.servebox.org/sites/actionscript-foundry/asapidoc/index.html
Lien41 : http://www.eclipse.org/
Lien42 : http://www.adobe.com/products/flex/
Lien43 : http://java.sun.com/javase/downloads/index_jdk5.jsp
Lien44 : http://servebox.developpez.com/tutoriels/as3/actionscript-foundry/gestionnaire-contacts-partie1/sources/apache-tomcat-6.0.16.zip
Lien45 : http://servebox.developpez.com/tutoriels/as3/actionscript-foundry/gestionnaire-contacts-partie1/sources/contact-application-source-partie1-debut.zip
Lien46 : http://www.servebox.org/actionscript-foundry/actionscript-foundry-documentation/a-mvc-framework/
Lien47 : http://www.servebox.org/actionscript-foundry/actionscript-foundry-documentation/remote-procedure-call/
Lien48 : http://www.adobe.com/devnet/flex/quickstart/using_data_binding/
Lien49 : http://servebox.developpez.com/tutoriels/as3/actionscript-foundry/gestionnaire-contacts-partie1/concept-de-base/
Lien50 : http://loic-joly.developpez.com/tutoriels/cpp/smart-pointers/
Lien51 : http://khayyam.developpez.com/articles/cpp/boost/serialization/
Lien52 : http://broux.developpez.com/articles/c/sockets/
Lien53 : http://hautrive.developpez.com/reseaux/
Lien54 : http://melem.developpez.com/reseaux/winsock/
Lien55 : http://arb.developpez.com/c++/boost/install/vc++/
Lien56 : http://ram-0000.developpez.com/tutoriels/cpp/boost-regex/#L2
Lien57 : http://www.developpez.com/
Lien58 : http://gwenael-dunand.developpez.com/tutoriels/cpp/boost/asio/
Lien59 : http://msdn.microsoft.com/en-us/library/ms536496(VS.85).aspx
Lien60 : http://bbil.ftp-developpez.com/tutoriel/vbs/interface-hta/fichiers/BonjourLeMonde.zip
Lien61 : http://bbil.ftp-developpez.com/tutoriel/vbs/interface-hta/fichiers/boutons.zip
Lien62 : http://bbil.ftp-developpez.com/tutoriel/vbs/interface-hta/fichiers/controles.zip
Lien63 : http://bbil.ftp-developpez.com/tutoriel/vbs/interface-hta/fichiers/Timer.zip
Lien64 : http://vb.developpez.com/faqvbs/?page=II.2
Lien65 : http://bbil.ftp-developpez.com/tutoriel/vbs/interface-hta/fichiers/ScriptingRuntime.zip
Lien66 : http://vb.developpez.com/faqvbs/?page=II.7
Lien67 : http://vb.developpez.com/faqvbs/?page=II.7#ctrlaltsuppr
Lien68 : http://bbil.ftp-developpez.com/tutoriel/vbs/interface-hta/fichiers/AccesRegistre.zip
Lien69 : http://vb.developpez.com/faqvbs/?page=III
Lien70 : http://bbil.ftp-developpez.com/tutoriel/vbs/interface-hta/fichiers/Wmi-partage.zip
Lien71 : http://bbil.developpez.com/tutoriel/vbs/interface-hta/

Lien72 : http://www.amazon.com/Cost-Based-Oracle-Fundamentals-Experts-Voice/dp/1590596366
Lien73 : http://www.oracle.com/technology/sample_code/tech/sql_plus/htdocs/demobld.html
Lien74 : http://marius-nitu.developpez.com/tutoriels/oracle/optimiseur/comment-optimiseur-oracle-calcule-cout/
Lien75 : http://fadace.developpez.com/ase/autoextension
Lien76 : http://fadace.developpez.com/sql/bitmask/
Lien77 : http://fadace.developpez.com/poweramc/
Lien78 : http://infocenter.sybase.com/help/topic/com.sybase.dc70204_1500/pdf/ase15pst.pdf
Lien79 : http://sybooks.sybase.com/nav/summary.do?prod=9938&lang=en&Submit.x=22&Submit.y=12&Submit=Submit&prodName=Adaptive+Server+Enterprise&archive=0
Lien80 : http://fadace.developpez.com/ase/reduction/
Lien81 : http://fr.wikipedia.org/wiki/Disque_dur
Lien82 : http://technet.microsoft.com/en-us/library/cc966412.aspx
Lien83 : http://www.amazon.fr/Managing-Maintaining-Microsoft-Windows-Environment/dp/0735622892/ref=sr_1_1?ie=UTF8&s=english-books&qid=1235210578&sr=8-1
Lien84 : http://mikedavem.developpez.com/sqlserver/tutoriels/architecture/
Lien85 : http://conception.developpez.com/reportage/agilite/xp-day-suisse-2009/
Lien86 : http://conception.developpez.com/livres/
Lien87 : http://irmatden.developpez.com/tutoriels/qt/integration-ogre-qt/
Lien88 : ftp://ftp-developpez.com/irmatden/tutoriels/fichiers/qt_ogre_select.zip
Lien89 : http://come-david.developpez.com/tutoriels/dps/?page=Strategie#LVIII
Lien90 : ftp://ftp-developpez.com/irmatden/tutoriels/fichiers/qt_ogre_newevent_camera.zip
Lien91 : http://doc.trolltech.com/4.4/qactiongroup.html
Lien92 : ftp://ftp-developpez.com/irmatden/tutoriels/fichiers/qt_ogre_newevent_selnode.zip
Lien93 : http://dico.developpez.com/html/3161-Conception-Command-design-pattern-command.php
Lien94 : http://www.vincehuston.org/dp/command.html
Lien95 : http://doc.trolltech.com/4.5/qundocommand.html
Lien96 : http://doc.trolltech.com/4.5/qundostack.html
Lien97 : http://doc.trolltech.com/4.5/qundoview.html
Lien98 : http://doc.trolltech.com/4.5/qundogroup.html
Lien99 : http://irmatden.developpez.com/tutoriels/qt/integration-ogre-qt-part-2/