



Developpez

Le Mag

Édition de octobre - novembre 2012.

Numéro 42.

Magazine en ligne gratuit.

Diffusion de copies conformes à l'original autorisée.

Réalisation : Alexandre Pottiez

Rédaction : la rédaction de Developpez

Contact : magazine@redaction-developpez.com

Sommaire

| | |
|------------|---------|
| JavaScript | Page 2 |
| (X)HTML | Page 4 |
| CSS | Page 10 |
| Lazarus | Page 16 |
| SAS | Page 23 |
| Access | Page 28 |
| C++ | Page 37 |
| Qt | Page 47 |
| 2D/3D/Jeux | Page 52 |
| Java | Page 55 |
| NetBeans | Page 63 |
| Eclipse | Page 64 |
| Android | Page 67 |
| Liens | Page 69 |

Article (X)HTML



Réaliser une application offline en HTML5

Cet article vous permettra de faire rapidement une application HTML5 qui gèrera le mode offline de votre navigateur.

par **Franck Lefevre**

Page [4](#)

Article Qt



Les modules de Qt 5

Qt 5 réorganise vos modules, retrouvez-les rapidement et facilement grâce à cet article.

par **Guillaume Belz**

Page [47](#)

Éditorial

Ce mois-ci, profitez de deux nouvelles rubriques totalement inédites dans le magazine, Lazarus et SAS.

Et bien entendu, retrouvez le meilleur des ressources de vos rubriques préférées.

Profitez-en bien !

La rédaction

HTML5 : quelques nouveautés de l'API DOM pour JavaScript

La spécification HTML5 définit différents modules indépendants. Cette modularité a pour avantage de permettre de travailler sur certains aspects du standard sans avoir besoin de se soucier de l'état d'avancement des autres.

Parmi ces modules, l'API DOM est celui qui permet de définir les propriétés et méthodes disponibles en JavaScript pour manipuler le DOM.

Nous allons voir les différentes nouveautés particulièrement utiles de cette API.

Standardisation de innerHTML

Introduite par Internet Explorer 4 (de mémoire...), la propriété **innerHTML** permet de définir le contenu d'un élément en lui affectant sous forme de chaîne une portion de code HTML.

```
HTMLElement.innerHTML = '<p>chaîne HTML</p>'
```

Notez qu'il s'agit d'une propriété et non d'une méthode, on l'utilise donc sous forme d'une affectation et non comme un appel de fonction.

Bien que propriétaire IE, cette propriété (sic) s'est largement répandue sur les autres navigateurs et est depuis longtemps largement compatible.

Elle est désormais standardisée (innerHTML : Lien [01](#)) et fait pleinement partie de la spécification HTML5.

getElementsByClassName()

Nous connaissons déjà les classiques *getElementById()*, *getElementsByName()* et *getElementsByTagName()*, mais beaucoup de développeurs ont longtemps regretté l'absence de possibilité de recherche d'éléments à partir d'un nom de classe (ce qui amenait souvent à utiliser une bibliothèque JavaScript pour pallier ce manque).

C'est désormais rectifié avec l'apparition de la méthode **getElementsByClassName()**.

Cette méthode peut être appliquée à l'objet *document* ou à un objet *HTMLElement*. Comme le 's' de 'elements' le laisse supposer, elle retourne une collection d'éléments (il est donc nécessaire de faire une boucle sur la collection pour traiter individuellement chaque élément).

Il est possible de spécifier plusieurs noms de classe pour un élément séparés par des espaces, la méthode prend bien entendu en compte cette subtilité.

```
var coll = document.getElementsByClassName('classe');
var coll = document.getElementById('un_id').getElementsByClassName('classe');
```

Cette méthode est accessible sur tous les navigateurs récents, y compris Internet Explorer depuis sa version 8.

querySelector() / querySelectorAll()

Ces deux nouvelles méthodes font leur apparition avec un fonctionnement un peu différent des précédentes.

Les méthodes du type *getElement[s]By...* prennent comme argument une chaîne correspondant au type d'élément recherché (un id, un nom de balise, de classe ou un name).

Les méthodes **querySelector()** et **querySelectorAll()** prennent quant à elles en argument une chaîne qui correspond à un sélecteur CSS.

Normalement, toute chaîne pouvant servir de sélecteur en CSS est acceptée, y compris les plus complexes. Nous ne listerons pas ici les différents sélecteurs possibles, je vous laisse faire vos propres tests.

Ces méthodes s'appliquent soit à l'objet *document*, soit à un objet *HTMLElement*.

querySelector() renvoie un élément HTML qui correspond au premier élément trouvé dans le DOM correspondant au sélecteur, son résultat peut donc être exploité directement.

querySelectorAll() renvoie une collection d'éléments HTML correspondant à tous les éléments correspondants au sélecteur. Pour traiter son résultat, il faudra donc faire une boucle sur chacun de ses membres.

```
var complexSelection = document.querySelector('.aChoisir ul[title="Liste à puce"], li &gt; a[href^=www]);
var allImportantDivInMyParagraph = document.getElementById('My').querySelectorAll('div.important');
```

Ces méthodes sont disponibles sur tous les navigateurs modernes y compris Internet Explorer depuis la version 9.

matchesSelector()

Cette méthode permet de vérifier si un élément correspond à un sélecteur CSS. Tout comme les précédentes, elle prend en argument une chaîne correspondant à un sélecteur. Elle renvoie un booléen.

```
var e1 = document.getElementById('elem1');
var e2 = document.getElementsByTagName('div')[0];
if(e1.matchesSelector('h1')){
    // traitement si l'élément est un titre de niveau 1
}
if(e2.matchesSelector('#toto')){
    // traitement si la première div du document possède l'id "toto"
}
```

Cette méthode est disponible sur tous les navigateurs récents y compris Internet Explorer depuis la version 9, cependant, seules les versions préfixées sont reconnues : *mozMatchesSelector()*, *webkitMatchesSelector()*, *oMatchesSelector()* et *msMatchesSelector()*.

L'interface classList

Jusqu'à présent, pour manipuler les classes CSS d'un élément, il était nécessaire d'utiliser des méthodes assez complexes, que ce soit pour vérifier la présence d'un nom de classe, pour en ajouter ou en retirer. Il fallait utiliser le plus souvent des expressions régulières et se méfier des erreurs possibles (typiquement, qu'une recherche sur « classe » ne renvoie pas vrai si l'élément possède la classe « classeN »).

L'interface **classList** permet de simplifier tout cela.

Elle s'applique à un élément HTML et renvoie une collection de ses noms de classe.

```
var allClasses = HTMLCollection.prototype.classList;
```

Elle dispose aussi de plusieurs méthodes permettant de manipuler cette liste de classes :

- **add()** permet d'ajouter un nom de classe à l'élément ;
- **remove()** permet de retirer un nom de classe à l'élément ;
- **toggle()** permet d'ajouter un nom de classe si

l'élément ne la possède pas, à le retirer sinon ;

- **contains()** permet de vérifier l'existence d'un nom de classe parmi celles de l'élément.

Cette interface est disponible sur tous les navigateurs modernes à l'exception d'Internet Explorer avant la version 10.

Conclusion

Ces différentes propriétés et méthodes apportent beaucoup aux développeurs pour ce qui est de la manipulation du DOM. Cependant, il me semble que de nombreuses méthodes utiles ont été oubliées, que ce soit en rapport à l'insertion d'éléments dans le DOM, à la possibilité de faire des recherches en remontant l'arborescence (retrouver facilement un élément à partir d'un de ses descendants). Espérons que les futures implémentations combleront ces manques en s'inspirant, par exemple, des différentes bibliothèques JavaScript.

Retrouvez ce billet blog de Didier Mouonval en ligne : [Lien 02](#)

Réaliser une application offline en HTML5

Cet article vous permettra de faire rapidement une application HTML5 qui gèrera le mode offline de votre navigateur.

1. Traduction

Cet article va vous apprendre à gérer trois technologies : HTML5 gérant le mode offline, le cache du navigateur et le stockage local. Notre exemple est une gestion de « to-do list » qui utilise trois technologies : HTML5 et JavaScript/jQuery pour le côté client et PHP pour le côté serveur.

2. Prérequis

Le code est simple. La page HTML contient une liste non numérotée contenant les tâches, ainsi qu'un formulaire. Dès que celui-ci est activé, le JavaScript intercepte le formulaire et l'envoi au fichier PHP pour qu'il l'enregistre dans un fichier texte.

Code HTML :

```
index.html
<!DOCTYPE html>
<html lang="fr">
<head>
  <title>Article sur HTML5 et le mode
  offline</title>
  <meta charset="utf-8">
  <link href="style.css" media="screen"
  rel="stylesheet" type="text/css">
  <script
  src="https://ajax.googleapis.com/ajax/libs/jquery
  /1.8.1/jquery.min.js"
  type="text/javascript"></script>
  <script src="script.js"
  type="text/javascript"></script>
</head>
<body>
  <div class="content">
    <h1>Article sur HTML5 et le mode offline</h1>
    <form method="post" action="index.html"
    id="frmTodo">
      <div>
        <label for="txtItem">Item :</label>
        <input type="text" name="txtItem"
        id="txtItem">
      </div>
      <input type="submit" value="Ajouter" />
    </form>

    <ul></ul>

    <p>Source : <br/><a
    href="http://icones.pro/supprimer-15-image-
    png.html" target="_blank">Image : </a></p>
  </div>
</body>
</html>
```

Code CSS :

```
style.css
body{
  background-color: #E0E0E0;
  color: #556677;
  font-family: Arial;
}
.content{
  margin:0 auto;
  text-align: center;
  width:300px;
}
form > div{
  padding: 10px;
}
input[type="text"]{
  border: 1px solid #556677;
  color: #556677;
  width: 150px;
}
input[type="submit"]{
  border: 1px solid #556677;
  color: #556677;
  font-weight: bold;
  padding: 5px 26px;
}
ul{
  padding: 0;
  text-align: left;
}
li{
  background-color: #CCCCCC;
  list-style-type: none;
  margin: 5px;
  padding: 5px;
}
li:hover{
  background-color: #AAAAAA;
}
li > img{
  cursor:pointer;
  float: right;
  height: 16px;
  width: 16px;
}
p{
  margin: 60px;
}
p > a{
  color: #556677;
  font-weight: bold;
  text-decoration: none;
}
```

```
p > a > img{
  height: 16px;
  width: 16px;
}
```

Code JavaScript :

script.js

```
$(document).ready(function() {
  var jqoTaskLi = $('<li><span
class="lblTask"></span></li>');

  // Lister les tâches à l'initialisation de la
page
$.ajax({
  url: 'ajax.php',
  dataType: 'json',
  type: 'POST',
  data: {
    type: 'list'
  },
  success: function(data) {
    if(data.result == true){
      $.each(data.data, function(key, value) {
        var jqoItem = jqoTaskLi.clone();
        jqoItem.find('.lblTask').html(value);
        $('ul').append(jqoItem);
      });
    }
  }
});

// Événement lors de la soumission du
formulaire
$('#frmTodo').submit(function() {
  // On récupère le texte nettoyé des espaces
avant et après
  var sVal = $('#txtItem').val();
  var sVal = $.trim(sVal);
  if(sVal != ''){
    // On envoie une requête AJAX de type POST
pour ajouter la tâche
    $.ajax({
      url: 'ajax.php',
      dataType: 'json',
      type: 'POST',
      data: {
        type: 'add',
        item: sVal
      },
      success: function(data) {
        if(data.result == true){
          var jqoItem = jqoTaskLi.clone();
          jqoItem.find('.lblTask').html(valu
e);

          $('ul').append(jqoItem);
          $('#txtItem').val('');
        }
      }
    });
  }
  return false;
});

$('#ulTodo').on('click', '.delTask', function() {
  var jqoParent = $(this).parent();
  // On envoie une requête AJAX de type POST
pour supprimer la tâche
  $.ajax({
```

```
url: 'ajax.php',
dataType: 'json',
type: 'POST',
data: {
  type: 'del',
  pos: $('li').index(jqoParent)
},
success: function(data) {
  if(data.result == true){
    jqoParent.remove();
  }
}
});
});
```

Code PHP :

ajax.php

```
<?php

$arrAction = array('add', 'del', 'list');
$arrResult = array('result' => false);

if(isset($_POST['type']) &&
in_array($_POST['type'], $arrAction)){
  // Si le fichier contenant les tâches
n'existe pas, on le crée.
  if(!file_exists('db.txt')){
    file_put_contents('db.txt', '');
  }

  // On charge le fichier et décode la chaîne
JSON contenant les tâches.
  $sDbTasks = file_get_contents('db.txt');
  $arrTasks = json_decode($sDbTasks);

  // Si l'action est un ajout...
  if($_POST['type'] == 'add'){
    // ... on ajoute l'élément aux tableaux des
tâches.
    $arrTasks[] = htmlentities($_POST['item'],
ENT_QUOTES);
    $arrResult['result'] = true;
  }

  // Si l'action est une suppression...
  elseif($_POST['type'] == 'del'){
    // ... et qu'il existe dans le tableau, on le
supprime
    if(is_numeric($_POST['pos'])){
      if(isset($arrTasks[$_POST['pos']])){
        array_splice($arrTasks, $_POST['pos'], 1);
        $arrResult['result'] = true;
      } else {
        $arrResult['result'] = false;
      }
    } else {
      $arrResult['result'] = false;
    }
  }

  // Si l'action est un listing...
  elseif($_POST['type'] == 'list'){
    $arrResult['result'] = true;
    $arrResult['data'] = $arrTasks;
  }

  // On encode le tableau des tâches dans une
```

```

chaîne JSON et on sauvegarde.
    $sDbTasks = json_encode($arrTasks);
    file_put_contents('db.txt', $sDbTasks);
}

echo json_encode($arrResult);
?>

```

3. La gestion du cache

3.1. Théorie

Le mode hors-ligne intervient quand votre ordinateur n'a plus de réseau ou que votre téléphone se situe dans une zone sans réseau. À ce moment-là, votre application Web utilise le cache du navigateur pour stocker les fichiers dont il aura besoin pour travailler (images, JavaScript, feuilles de style et pages HTML). Ces fichiers seront déclarés dans un fichier de type MANIFEST lié à votre fichier HTML via la balise HTML comme ci-dessous :

```

index.html
<html manifest="filename.manifest">

```

Pour que ce fichier soit bien déclaré au niveau du serveur, il nous faut déclarer son MIME-TYPE via un fichier .htaccess :

```

.htaccess
AddType text/cache-manifest manifest

```

Après que le fichier a bien été déclaré au niveau du HTML et du serveur, nous allons voir comment il est structuré :

```

manifest.manifest
CACHE MANIFEST
# Comment

CACHE:
delete.png
index.html
script.js
style.css
# Fichier JS : jQuery
https://ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js

FALLBACK:

NETWORK:
/ajax.php

```

Ce fichier MANIFEST est divisé en quatre parties :

- l'en-tête qui doit être déclaré comme tel : CACHE MANIFEST ;
- la zone de CACHE contenant les fichiers statiques à mettre dans le cache du navigateur ;
- la zone de FALLBACK contenant la redirection à mettre en place pour le cas où ils ne seraient pas accessibles en ligne ;
- la zone de NETWORK contenant les fichiers nécessitant une connexion internet (par exemple, les pages de connexion ou les fichiers PHP liés à de l'AJAX).

Comme vous pouvez le voir, un commentaire peut être placé à n'importe quel endroit du fichier si un dièse est placé au début de la ligne.

3.2. Pratique

Après la théorie, la pratique. Nous mettons donc en place le fichier manifest, le .htaccess et la modification de la balise HTML. En visitant la page sous Firefox, un message s'affiche dans le haut du navigateur (figure 3.2.1).



Figure 3.2.1 : Mozilla Firefox propose le téléchargement du cache

Après autorisation, on active le travail hors connexion de Mozilla Firefox.

Actualisons la page.

Résultat : la page HTML et les contenus statiques sont là. Maintenant, testons un ajout et une suppression de tâche : cela fonctionne.

4. Le mode hors-ligne

4.1. Test

On active le travail hors connexion de Mozilla Firefox. Comme on peut le voir dans la figure IV.1.1, la requête AJAX vers le fichier ajax.php a échoué.

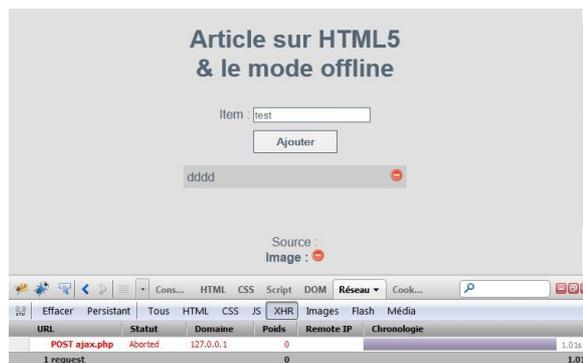


Figure IV.1.1 : Échec de la requête AJAX

4.2. Théorie

Dans ce cas, nous devons détecter quand il n'y a plus de réseau pour empêcher les requêtes AJAX, et quand le réseau est de nouveau disponible, permettre les requêtes vers le serveur. Pour cela, on utilise la propriété onLine de l'objet window.navigator (référence : Lien 03) qui retourne un booléen contenant le statut en ligne du navigateur. De plus, pour voir les changements réseaux, deux événements existent : window.onOnline et window.onOffline.

4.3. Pratique

Du côté pratique, il suffit d'interroger la variable

navigator.onLine qui contient le statut du navigateur et de suivre via callbacks les deux événements window.onOnline et window.onOffline.

script.js

```
$(document).ready(function() {
    var jqoTaskLi = $('<li><span
class="lblTask"></span></li>');

    // Lister les tâches à l'initialisation de la
page
$.ajax({
    url: 'ajax.php',
    dataType: 'json',
    type: 'POST',
    data: {
        type: 'list'
    },
    success: function(data) {
        if(data.result == true) {
            $.each(data.data, function(key, value) {
                var jqoItem = jqoTaskLi.clone();
                jqoItem.find('.lblTask').html(value);
                $('ul').append(jqoItem);
            });
        }
    }
});

// Événements lors de la mise on/offline du
navigateur
$(document).on('online', function() {
});
$(document).on('offline', function() {
    alert('Vous êtes maintenant hors-ligne. Vos
tâches ne seront pas enregistrées');
});

// Événement lors de la soumission du
formulaire
$('#frmTodo').submit(function() {
    // On récupère le texte nettoyé des espaces
avant et après
    var sVal = $('#txtItem').val();
    var sVal = $.trim(sVal);
    // Si on est en ligne
    if(navigator.onLine) {
        if(sVal != '') {
            // On envoie une requête AJAX de type
POST pour ajouter la tâche
            $.ajax({
                url: 'ajax.php',
                dataType: 'json',
                type: 'POST',
                data: {
                    type: 'add',
                    item: sVal
                },
                success: function(data) {
                    if(data.result == true) {
                        var jqoItem = jqoTaskLi.clone();
                        jqoItem.find('.lblTask').html(sVal);
                        $('ul').append(jqoItem);
                        $('#txtItem').val('');
                    }
                }
            });
        }
    }
});
});
```

```
// Si on n'est pas en ligne
else {
    alert('Réseau : Offline');
}
return false;
});
$('#ulTodo').on('click', '.delTask', function() {
    var jqoParent = $(this).parent();
    // Si on est en ligne
    if(navigator.onLine == true) {
        // On envoie une requête AJAX de type POST
pour supprimer la tâche
        $.ajax({
            url: 'ajax.php',
            dataType: 'json',
            type: 'POST',
            data: {
                type: 'del',
                pos: $('li').index(jqoParent)
            },
            success: function(data) {
                if(data.result == true) {
                    jqoParent.remove();
                }
            }
        });
    }
    // Si on n'est pas en ligne
    else {
        alert('Réseau : Offline');
    }
});
});
```

5. Le stockage local

À ce niveau, les tâches ne sont envoyées uniquement que dans le cas où le navigateur est en ligne. Si le navigateur est hors ligne, un simple message d'erreur nous prévient. Mais comme la technologie HTML5 est bien conçue : nous allons pouvoir stocker les actions via la technologie du stockage local, plus communément nommé localStorage.

5.1. Théorie

Le stockage local permet de stocker facilement des informations de type clé-valeur. L'objet localStorage contient six méthodes :

- localStorage.clear() : supprime toutes les clés stockées ;
- localStorage.getItem(sKey) : retourne la valeur associée à une clé ;
- localStorage.key(iIndex) : retourne la clé d'un index ;
- localStorage.length : retourne le nombre de clés stockées ;
- localStorage.removeItem(sKey) : supprime une clé et sa valeur associée ;
- localStorage.setItem(sKey, sValue) : stocke une clé avec une valeur associée.

5.2. Pratique

Dans notre projet, nous allons avoir besoin du stockage local à deux moments :

- en cas de navigateur hors ligne pour stocker les

actions (ajout et suppression) ;

- en cas de retour au mode en ligne pour lister et exécuter les actions stockées précédemment.

script.js

```
jQuery(function($) {
    var jqoTaskLi = $('<li><span
class="lblTask"></span></li>');

    // Lister les tâches à l'initialisation de la
page
$.ajax({
    url: 'ajax.php',
    dataType: 'json',
    type: 'POST',
    data: {
        type: 'list'
    },
    success: function(data) {
        if(data.result == true){
            $.each(data.data, function(key, value) {
                var jqoItem = jqoTaskLi.clone();
                jqoItem.find('.lblTask').html(value);
                $('ul').append(jqoItem);
            });
        }
    }
});

    // Événements lors de la mise on/offline du
navigateur
$(document).on('online', function() {
    if(localStorage.getItem('numActions')==null){
        localStorage.setItem('numActions', 0);
    }
    var numActions =
localStorage.getItem('numActions');
    numActions = parseInt(numActions);
    if(numActions > 0){
        // On parse toutes les actions
        for (iInc = 0; iInc < numActions; iInc++) {
            var sAction =
localStorage.getItem('action'+iInc);
            // On dirige en fonction des actions
            if(sAction == 'add'){
                // On envoie la requête AJAX d'ajout
                $.ajax({
                    url: 'ajax.php',
                    dataType: 'json',
                    type: 'POST',
                    data: {
                        type: 'add',
                        item: localStorage.getItem('item'+iInc)
                    },
                    success: function(data) {
                        if(data.result == false){
                            alert('Error')
                        }
                    }
                });
            }
            else if(sAction == 'del'){
                // On envoie la requête AJAX de
suppression
                $.ajax({
                    url: 'ajax.php',
```

```
                    type: 'POST',
                    data: {
                        type: 'del',
                        pos: localStorage.getItem('pos'+iInc)
                    },
                    success: function(data) {
                        if(data.result == false){
                            alert('Error')
                        }
                    }
                });
            }
        }
        // On nettoie toutes les actions
        localStorage.clear();
    }
});
$(document).on('online', function() {
    localStorage.setItem('numActions', 0);
});

    // Événement lors de la soumission du
formulaire
$('#frmTodo').submit(function(){
    var sVal = $('#txtItem').val();
    var sVal = $.trim(sVal);

    // Si on est en ligne
    if(navigator.onLine){
        // On récupère le texte nettoyé des espaces
avant et après
        if(sVal != ''){
            // On envoie une requête AJAX de type
POST pour ajouter la tâche
            $.ajax({
                url: 'ajax.php',
                dataType: 'json',
                type: 'POST',
                data: {
                    type: 'add',
                    item: sVal
                },
                success: function(data) {
                    if(data.result == true){
                        var jqoItem = jqoTaskLi.clone();
                        jqoItem.find('.lblTask').html(sVal);
                        $('ul').append(jqoItem);
                        $('#txtItem').val('');
                    }
                }
            });
        }
        // Si on n'est pas en ligne
        else {
            if(localStorage.getItem('numActions') ==
null){
                localStorage.setItem('numActions', 0);
            }
            var numActions =
localStorage.getItem('numActions');
            numActions = parseInt(numActions);
            // On enregistre en local
            localStorage.setItem('action'+numActions,
'add');
            localStorage.setItem('item'+numActions,
sVal);
            localStorage.setItem('numActions',
numActions + 1);
            // On exécute l'action comme si on était en
```

```

ligne
    var jqoItem = jqoTaskLi.clone();
    jqoItem.find('.lblTask').html(sVal);
    $('#ul').append(jqoItem);
    $('#txtItem').val('');
}
return false;
});
$('#ulTodo').on('click', '.delTask', function()
{
    var jqoParent = $(this).parent();
    // Si on est en ligne
    if(navigator.onLine == true){
        // On envoie une requête AJAX de type POST
pour supprimer la tâche
        $.ajax({
            url: 'ajax.php',
            dataType: 'json',
            type: 'POST',
            data:{
                type:'del',
                pos:$('li').index(jqoParent)
            },
            success: function(data) {
                if(data.result == true){
                    jqoParent.remove();
                }
            }
        });
    }
    // Si on n'est pas en ligne
    else {
        if(localStorage.getItem('numActions') ==
null){
            localStorage.setItem('numActions', 0);

```

```

}
    var numActions =
localStorage.getItem('numActions');
    numActions = parseInt(numActions);
    // On enregistre en local
    localStorage.setItem('action'+numActions,
'del');
    localStorage.setItem('pos'+numActions, $
('li').index(jqoParent));
    localStorage.setItem('numActions',
numActions + 1);
    // On exécute l'action comme si on était en
ligne
    jqoParent.remove();
}
});
});

```

6. Conclusion

Et voilà, vous savez désormais gérer la déconnexion réseau, le cache de votre application Web et même du stockage local. Ce code est améliorable sur de nombreux points dont voici quelques pistes :

- **sécuriser** la gestion des erreurs avec par exemple, la protection des failles XSS (essayer d'ajouter la tâche `<script>alert('Test');</script>`);
- créer une image qui sera modifiée à chaque changement de statut réseau.

Vous pouvez retrouver toutes les sources sur mon compte GitHub dédié aux articles Développez.com : Lien [04](#)

Retrouvez l'article de Franck Lefevre en ligne : [Lien 05](#)

Les derniers tutoriels et articles

Les bordures en CSS3

Il n'est peut-être plus la peine de vous présenter les avantages et nouveautés apportées par la troisième version du CSS (Cascading Style Sheets). L'un des grands pas dans ce langage est l'apparition de propriétés de mise en forme avancée des bordures.

1. Introduction

Nous verrons ici comment créer des bordures aux coins arrondis (`border-radius`), des bordures à partir d'image (`border-image`), des bordures aux couleurs dégradées (`border-color`) et enfin, des ombres portées aux boîtes (`box-shadow`). Et un petit bonus que je vous laisse découvrir par vous-même !

2. Il n'y a pas que les footballeurs qui réussissent leurs corners

2.1. Généralités

Jusqu'à présent pour réaliser des boîtes aux coins arrondis, nous devons utiliser une ou plusieurs images à appliquer en arrière-plan dans des `<div>` imbriqués ou des tableaux à neuf cellules. Nous passons plus de temps à découper nos éléments en un coin en haut à gauche, puis à droite, recoller les morceaux parce qu'on a donné un coup de ciseau de travers... Eh bien tout ceci est terminé ! Pour peu que le navigateur soit assez récent pour comprendre ce que nous allons lui demander. IE - avant sa version 9 - ne prend pas en charge les propriétés qui suivent.

Avec le CSS3, il nous est possible de faire un angle arrondi via la propriété `border-radius`.

Voici sa syntaxe : `<valeur>{1,4} / <valeur>{1,4}`

Sa compatibilité : IE9+, Firefox 4+, Chrome, Safari 5+ et Opera 10.50+.

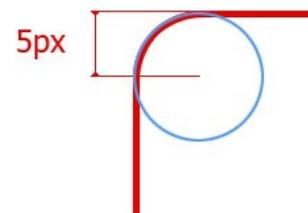
```
.corner {
  border-radius: 5px;
}
```

```
<div style="background-color: red; width: 150px;
height: 150px; color: white;" class="corner">Ceci
n'est pas un carton rouge !</div>
```



Vous voyez, rien de plus simple. Ici, tous les coins ont le même arrondi.

Nous définissons une valeur pour le rayon du cercle décrivant la forme de l'arrondi. Voici un schéma d'illustration :



Il est bien évidemment possible de spécifier des valeurs différentes pour chacun des coins de l'élément.

Pour rappel : la syntaxe de cette propriété nous permet de mettre de une à quatre valeurs qui peuvent être d'unités diverses (px, em, %) :

- une valeur : pour les quatre côtés ;
- deux valeurs : la première pour le haut et le bas, la seconde pour la gauche et la droite ;
- trois valeurs : la première pour le haut, la seconde pour la gauche et la droite, la troisième pour le bas ;
- quatre valeurs : respectivement haut, droite, bas et gauche.

```
.corner {
  -moz-border-radius: 5px 0 50px 10px;
  -webkit-border-radius: 5px 0 50px 10px;
  -khtml-border-radius: 5px 0 50px 10px;
  border-radius: 5px 0 50px 10px;
}
```

```
<div style="background-color: red; width: 200px;
height: 100px;" class="corner"></div>
```



Heu...m'sieur l'arbitre ! C'est qui -moz, -webkit et -khtml ?

Outre le fait que nous avons défini des arrondis plus ou moins importants sur les différents angles de notre div, vous avez pu remarquer que des « nouvelles » propriétés se sont invitées dans la partie.

Il s'agit de propriétés propriétaires spécifiques à chaque navigateur. Cela leur permet d'implémenter et de tester des

propriétés qui ne sont pas encore recommandées par le W3C.

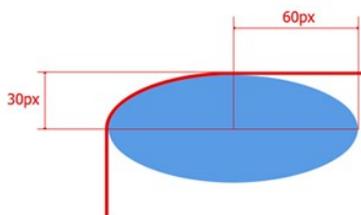
Les avis divergent sur leurs utilisations. Certains encouragent à les omettre prétextant une implémentation de `border-radius` largement répandue depuis quelques versions dans certains navigateurs, d'autres préfèrent les laisser pour assurer une compatibilité accrue. Dans la suite de ce tutoriel, je n'emploierai que la propriété sans préfixe pour des raisons de lisibilité.

2.2. En détail

Maintenant que nous avons vu comment faire des arrondis, nous allons voir que nous pouvons pousser le bouchon de Maurice un peu plus loin.

En effet, enfin... les effets de `border-radius` peuvent être étendus en mettant des couples de valeurs. Ces couples dont les valeurs sont séparées par un slash (/) permettent de définir les rayons horizontaux pour la première valeur et les rayons verticaux pour la deuxième. La différence ici est que ces deux rayons s'appliquent sur une ellipse et non plus un cercle. C'est pourquoi les formes peuvent être diverses et variées.

Un autre schéma d'illustration pour mieux comprendre :



Reprenons notre exemple en appliquant une valeur pour l'horizontale et la verticale :

```
.corner {  
  border-radius: 10px 50px / 50px;  
}
```

```
<div style="background-color: red; width: 200px;  
height: 100px;" class="corner"></div>
```



Dans le cas présent, les coins en haut à gauche et en bas à droite sont définis par une ellipse avec des rayons verticaux et horizontaux différents (respectivement 50px et 10px).

Comme c'est souvent le cas en CSS, cet attribut peut être détaillé. C'est-à-dire que l'on ne peut arrondir que certains angles en les précisant spécifiquement.

```
.corner {  
  border-top-left-radius: 50px 10px;  
  border-bottom-right-radius: 25px;  
}
```

```
<div style="background-color: red; width: 200px;  
height: 100px;" class="corner"></div>
```



Vous l'aurez compris, seuls les angles en haut à gauche et en bas à droite sont paramétrés.

Voici les propriétés disponibles avec les équivalents pour les propriétés propriétaires au cas où vous deviez les utiliser. Les standards sans préfixe (ce qui est le cas pour Opera) :

- `border-radius` ;
- `border-top-left-radius` ;
- `border-top-right-radius` ;
- `border-bottom-right-radius` ;
- `border-bottom-left-radius`.

Safari et Chrome (pour ne citer qu'eux) :

- `-webkit-border-radius` ;
- `-webkit-border-top-left-radius` ;
- `-webkit-border-top-right-radius` ;
- `-webkit-border-bottom-right-radius` ;
- `-webkit-border-bottom-left-radius`.

Mozilla Firefox :

- `-moz-border-radius` ;
- `-moz-border-radius-topleft` ;
- `-moz-border-radius-topright` ;
- `-moz-border-radius-bottomright` ;
- `-moz-border-radius-bottomleft`.

2.3. S'il te plaît, dessine-moi un mouton !

Voici quelques exemples d'utilisation de `border-radius` :

```
#exemple div {  
  background-color: gray;  
  width:200px;  
  height: 80px;  
  float: left;  
  margin-left: 30px;  
  margin-top: 30px;  
  line-height: 80px;  
  text-align: center;  
  font-size: 25px;  
  color: white;  
  font-weight: bold;  
}  
  
#demo1 {  
  border-radius: 35px;  
}  
  
#demo2 {  
  border-top-left-radius: 30px 15px;  
}  
  
#demo3 {  
  border-bottom-left-radius: 25px 50px;  
}
```

```
#demo4 {
  border-radius: 25px 10px / 10px 25px;
}

#demo5 {
  border-radius: 1em 4em 1em 4em;
}

#demo6 {
  border-top-left-radius: 50px;
}
```

```
<div id="exemple">
  <div id="demo1"></div>
  <div id="demo2"></div>
  <div id="demo3"></div><p style="clear:
both;"></p>
  <div id="demo4"></div>
  <div id="demo5"></div>
  <div id="demo6"></div>
</div>
```



3. « L'image vaut mille mots. » - Confucius

Il est désormais possible de définir une image comme motif de bordure en lieu et place des traits dont nous disposons jusqu'à maintenant.

Notre image de référence :



3.1. Une balise unique pour les gouverner toutes

Border-image regroupe une liste de propriétés que nous verrons brièvement ci-dessous. Pourquoi brièvement ? Parce qu'elles ne sont pas encore reconnues par les navigateurs, à l'exception donc, de border-image dont l'exemple complet sera donné à la fin de la présentation des propriétés qu'elle regroupe.

Les propriétés présentes dans le point A ne sont pas reconnues par nos navigateurs. Elles ne sont données qu'à titre d'exemple et d'explication pour la propriété raccourcie : border-image.

3.1.1. Border-image-source

Permet de définir l'URL de l'image à utiliser. On peut utiliser les mêmes types d'images que pour les images de fond, à savoir : les jpeg, gif, png, svg, les dégradés, des images codées en base64.

```
.corner-image {
  border-image-source : url('fond.gif');
}
```

3.1.2. Border-image-slice

Permet de définir jusqu'à quatre longueurs indiquant la distance à partir de chaque coin de l'image afin de la couper.

Un petit schéma pour mieux comprendre :



Nous avons découpé notre image en neuf parties :

- 1, 3, 7 et 9 : ce sont nos quatre coins. Ils ne seront pas modifiés (ni étirés, ni répétés, ni même les deux) ;
- 2, 4, 5, 6, 8 : ces parties-là seront étirées, répétés ou les deux. Tout dépend de vos attentes.

Ici, pour cet exemple, le CSS ressemblerait à :

```
.corner-image {
  border-image-slice: 26 26 26 26;
  border-image-slice: 26 /* En version
raccourcie vu que les côtés sont tous de la même
longueur */
}
```

L'image est découpée en neuf carrés de 26 pixels de côté.

3.1.3. Border-image-width

Permet de définir la largeur de la bordure. Elle est prioritaire sur border-width. Dans le cas où aucune n'est présente, la bordure sera celle par défaut de border-width.

Opera ne prend pas en charge border-image-width, il faudra donc utiliser border-width à la place.

```
.corner-image {
  border-image-width: 26px;
}
```

3.1.4. Border-image-outset

La propriété border-image-outset indique de combien la zone de l'image peut s'étendre au-delà de la dimension de la bordure sur les quatre côtés.

```
.corner-image {
  border-image-outset: 26;
}
```

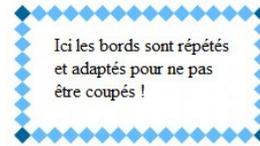
3.1.5. Border-image-repeat

Permet de définir de quelle manière seront traités les bords en haut, à droite, en bas et à gauche de l'image. Cela ne concerne pas les angles, bien évidemment puisqu'eux ne sont pas répétés. Il est possible de définir une ou deux valeurs, la première s'appliquant pour le haut et le bas, tandis que la deuxième sera pour la droite et la gauche. Si une seule est fournie elle s'applique sur les quatre côtés.

Les valeurs disponibles sont :

- stretch : étire l'image sur toute la longueur à couvrir ;
- repeat : comme pour background, l'image sera

- répétée pour couvrir tout l'espace nécessaire ;
- round : comme repeat, à la différence que cette fois, l'échelle de l'image sera conservée afin de ne plus avoir de coupure.



Le code ci-dessous montre leur mise en application.

```
.corner-image-1 {
  border-image-repeat: stretch stretch;
}

.corner-image-2 {
  border-image-repeat: repeat repeat;
}

.corner-image-3 {
  border-image-repeat: round round;
}
```



3.2. Enfin ! La voilà ! Border-image !

Alors pour faire court, vous prenez tout ce qu'on a fait précédemment et vous les mettez à la suite. Dans l'ordre énoncé.

Voici la syntaxe de border-image :<source> <slice {1,4}> / <width {1,4}> / <outset> <repeat {1,2}>

Sa compatibilité : Firefox avec le préfixe -moz, Opera avec le préfixe -o, Safari avec le préfixe -webkit. Internet Explorer ne prend pas encore en charge cette propriété. Chrome quant à lui supporte cette propriété.

Exemple :

```
.corner-image {
  border-width: 26px; /* Fix pour Opera */
  border-image: url('fond.gif') 26 / 26px
  round;
}
```

Qui est la version abrégée de :

```
.corner-image {
  border-width: 26px; /* Fix pour Opera */
  border-image: url('fond.gif') 26 26 26 26 /
  26px 26px 26px 26px round; /* Je vous rappelle
  que les quatre côtés sont égaux donc pas besoin
  de répéter leurs valeurs */
}
```

Nous avons donc une bordure constituée d'une image dont les parties du milieu sont répétées sans coupures (round).

4. Sortez vos feutres, c'est atelier coloriage

Grâce à la version trois de CSS, il est possible d'avoir des dégradés de couleurs dans nos bordures. Actuellement, seul Firefox a implémenté la gestion du dégradé. L'utilisation du préfixe -moz est donc obligatoire.

Voici la syntaxe de -moz-border-colors : [<color> || transparent]+

La compatibilité de border-colors : Firefox 3.0+

Un exemple d'utilisation :

```
.color-border-1 {
  border: 6px solid black;
  -moz-border-bottom-colors: #0682bd #198dc9
  #2d99d7 #3da3e1 #51afee #62baf9;
  -moz-border-top-colors: #0682bd #198dc9
  #2d99d7 #3da3e1 #51afee #62baf9;
  -moz-border-left-colors: #0682bd #198dc9
  #2d99d7 #3da3e1 #51afee #62baf9;
  -moz-border-right-colors: #0682bd #198dc9
  #2d99d7 #3da3e1 #51afee #62baf9;
}
```

```
<div style="background-color: red; width: 200px;
height: 100px;" class="color-border-1"></div>
```



On déclare une couleur par pixel. Donc si on a une bordure de six pixels, on déclare six couleurs.

5. L'ombre c'est pas qu'avec les parasols ?

La dernière (ou pas...) propriété que nous verrons aujourd'hui est box-shadow. Avec cette dernière, il vous est possible d'ajouter une ombre interne ou externe sur un élément HTML de type bloc. Pour le texte il faudra utiliser text-shadow qui est prévu à cet effet.

Voici la syntaxe de box-shadow : none | <shadow> [, <shadow>]*

avec : <shadow> = inset? && [<length>{2,4} && <color>?]

Sa compatibilité : IE 9+, Firefox 4+, Safari 4+, Opera 10.5+, Chrome 10+

Un petit peu d'explications ne serait pas du luxe :

- une valeur positive pour le décalage horizontal déplace l'ombre vers la droite, une valeur négative vers la gauche ;

- une valeur positive pour le décalage vertical déplace l'ombre vers le bas, une valeur négative vers le haut ;
- vous ne pouvez pas donner une valeur négative pour l'étendue de flou. Plus la valeur est grande, plus le flou s'étale ;
- une distance de propagation positive entraîne une expansion de la zone d'ombre dans toutes les directions, une valeur négative entraîne une contraction ;
- la couleur est celle de l'ombre ;
- la valeur inset indique une ombre interne plutôt qu'externe.

Je vous assure que cela va être plus clair avec un exemple, que voici d'ailleurs :

```
.shadow {
  box-shadow: 2px 2px 4px black;
}
```

```
<div style="background-color: yellow; width: 200px; height: 100px;" class="shadow"></div>
```



Ici, notre ombre est décalée de deux pixels horizontalement et verticalement avec un flou de quatre pixels, le tout de couleur noire.

Là, même chose avec des valeurs légèrement différentes :

```
.shadow {
  box-shadow: 20px 20px black;
}
```

```
<div style="background-color: yellow; width: 200px; height: 100px;" class="shadow"></div>
```



Vous comprenez maintenant comment donner plus de dimension à vos boîtes ?

Un autre exemple mais avec une ombre interne :

```
.shadow {
  box-shadow: inset 0 0 5px 5px #888;
}
```

```
<div style="background-color: yellow; width: 200px; height: 100px;" class="shadow"></div>
```

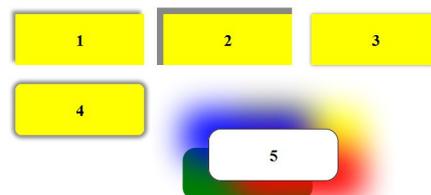


Je crois que vous avez compris le principe maintenant.

Voici d'autres exemples d'utilisation :

```
#exemple div {
  width:200px;
  height: 80px;
  float: left;
  line-height: 80px;
  text-align: center;
  font-size: 25px;
  color: black;
  font-weight: bold;
}
#demo1 {
  background-color: yellow;
  box-shadow: -5px -5px 5px #888;
  margin-left: 30px;
  margin-top: 30px;
}
#demo2 {
  background-color: yellow;
  box-shadow: -5px -5px 0 5px #888; /* On
propage la zone d'ombre de 5px */
  margin-left: 30px;
  margin-top: 30px;
}
#demo3 {
  background-color: yellow;
  box-shadow: 0 0 5px #888;
  margin-left: 30px;
  margin-top: 30px;
}
#demo4 {
  background-color: yellow;
  box-shadow: 0 0 5px 5px #888;
  margin-left: 30px;
  margin-top: 30px;
  border-radius: 10px;
}
#demo5 {
  background-color: white;
  border: 1px solid black;
  border-radius: 20px;
  margin: 100px;
  box-shadow: 40px 30px 50px red, -40px -30px
50px blue, 40px -30px 50px yellow, -40px 30px
green;
}
```

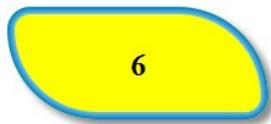
```
<div id="exemple">
  <div id="demo1">1</div>
  <div id="demo2">2</div>
  <div id="demo3">3</div><p style="clear:
both;"></p>
  <div id="demo4">4</div>
  <div id="demo5">5</div>
</div>
```



Petit exemple, juste comme ça, avec tout ce qui a été vu

jusqu'à maintenant :

```
#demo6 {
  background-color: yellow;
  box-shadow: 5px 5px 5px #888;
  margin-left: 30px;
  margin-top: 30px;
  border-radius: 1em 4em 1em 4em;
  border: 6px solid black;
  -moz-border-bottom-colors: #0682bd #198dc9
#2d99d7 #3da3e1 #51afee #62baf9;
  -moz-border-top-colors: #0682bd #198dc9
#2d99d7 #3da3e1 #51afee #62baf9;
  -moz-border-left-colors: #0682bd #198dc9
#2d99d7 #3da3e1 #51afee #62baf9;
  -moz-border-right-colors: #0682bd #198dc9
#2d99d7 #3da3e1 #51afee #62baf9;
}
```



On peut vraiment faire des choses intéressantes avec CSS3.

6. Bonus

Allez, une dernière propriété pour la route et après je vous laisse avec « Conclusion et remerciements », ils sont sympas vous verrez ;)

Je n'en ai pas parlé jusqu'à présent mais CSS3 offre une énième façon de déclarer une couleur pour un élément : la RGBA.

Ce n'est ni plus ni moins que le système RGB traditionnel complété du canal Alpha, qui permet de jouer avec l'opacité, la transparence d'une couleur. Et ça, c'est intéressant !

Par exemple : définissons une div bleue et une deuxième, également bleue, mais avec une transparence de 50 %.

```
body {
  background: gray;
}
article {
  background: rgb(255, 0, 0);
  width: 200px;
  height: 100px;
  position: relative;
}
.rgba-1 {
  width: 75px;
  height: 75px;
  background: rgba(0, 0, 255, 1);
  position: absolute;
}
.rgba-2 {
  width: 75px;
  height: 150px;
}
```

```
background: rgba(0, 0, 255, 0.5);
position: absolute;
left: 75px;
}
```

```
<article>
  <div class="rgba-1"></div>
  <div class="rgba-2"></div>
  Lorem ipsum dolor sit amet, consectetur
  adipiscing elit, sed do eiusmod tempor
  incididunt ut labore et dolore magna aliqua. Ut
  enim ad minim veniam, quis.
</article>
```



Vous remarquez que la deuxième div semble d'une couleur différente pourtant elle est juste transparente.

Les valeurs de la couche Alpha varient de 0 à 1, de la plus claire à la plus foncée.

Attention, le système RGBA n'est pas à confondre avec la propriété opacity. Elles n'ont pas les mêmes utilisations. Opacity rend l'élément entier transparent, contenu compris, alors que RGBA ne s'applique qu'aux couleurs.

Observez la différence :

```
.rgba {
  background: rgba(0, 125, 0, 0.5);
}
.opacity {
  background: rgb(0, 125, 0);
  opacity: 0.5;
}
```

```
<div style="width: 200px; height: 20px; line-height: 20px;" class="rgba">Ceci est un texte avec RGBA</div>
<div style="width: 200px; height: 20px; line-height: 20px;" class="opacity">Ceci est un texte avec Opacity</div>
```

Ceci est un texte avec RGBA

Ceci est un texte avec Opacity

On voit ici que dans le cas d'opacity, le texte est également mis en transparence.

7. Conclusion

Voilà, c'est tout pour l'instant. J'espère que ce petit article vous aura plu et que je vous retrouverai dans la suite des aventures du CSS3.

Retrouvez l'article de Torgar en ligne : [Lien 06](#)

GENÈSE D'UN DICTIONNAIRE - Construction d'un lexique interactif avec Lazarus

Cette série d'articles a pour objectif la construction d'un lexique interactif, chaque étape traitant les difficultés rencontrées dans la gestion du vocabulaire, et présente une ou des solutions.

1. Lecture d'une liste de mots

1.1. Introduction

La saisie assistée de SMS, l'accès aux moteurs de recherche et la correction automatique d'orthographe nous ont familiarisés avec un processus remarquable : en temps réel, un robot nous propose de compléter le mot que l'on a en tête...

La prestation est d'autant plus notable qu'elle s'est insinuée dans nos tâches quotidiennes sans crier gare : l'assistance est là, permanente, rassurante, quelquefois même envahissante ; ceci dans des conditions techniques — stockages de données monstrueux, flux d'échanges à débits élevés, algorithmes de tri rapide mieux gardés que Fort Knox, etc. — que l'on peut difficilement imaginer.

Il n'est pas question ici d'aborder cet arsenal à la pointe de l'informatique d'aujourd'hui, mais modestement d'étudier comment réaliser une base de vocabulaire personnelle, évolutive, consultable aisément, ce qui est déjà une approche intéressante, mais, pourquoi pas, susceptible de donner un coup de main dans le cadre d'un jeu de mots :

- recherche formelle avec l'utilisation d'un masque précisant la longueur du mot et les lettres connues,
- recherche logique à partir d'une définition souvent humoristique. Un exemple célèbre est celui de Tristan Bernard : « Vide les baignoires et remplit les lavabos ». Réponse... à la fin du chapitre.

En quelques étapes, nous allons construire une imposante base de mots avec les outils nécessaires pour la consulter et l'enrichir.

Oui, ça marche : mieux que Google !

L'approche se veut méthodique et progressive. Didactique. Les geeks ne trouveront rien de transcendant. Les débutants et les curieux pourront au contraire apprécier une méthode essentiellement pragmatique...

Et après tout, rien n'empêchera de porter l'application sur Android, pour que l'application, une fois achevée, soit accessible sur son PDA...

Les passionnés et les bonnes volontés pourront apporter leur compétence, qui sera la bienvenue.

1.2. Présentation

La première étape est consacrée au chargement d'une liste et à sa consultation. Les suivantes constituent une initiation ludique à la programmation sous Lazarus, mais elles restent transposables aisément à tout autre langage.

La saisie d'un vocabulaire de 300 000 mots est une tâche digne des copistes du Moyen Âge : à raison d'un mot par seconde, il faudrait quatre jours de frappe ininterrompue... Heureusement, avec internet, l'accès à ce volume de données est facile.

1.3. Liste de départ

Dans votre navigateur préféré, collez l'adresse suivante, ou cliquez tout simplement sur le lien : [Lien 07](#)

Sur la page qui apparaît, cliquez sur " liste.de.mots.francais.frgut.txt " ([Lien 08](#)).

Le navigateur s'ouvre sur une nouvelle page qui présente la liste recherchée. Un clic droit, dans le menu contextuel, choisissez **Enregistrer sous** ; vous pouvez conserver le nom proposé ; choisissez comme répertoire celui du projet, par exemple **Lexique**, et sauvegardez.

1.4. La source

Le fichier qui vient d'être chargé est un véritable trésor : il contient exactement 336 531 entrées, donc beaucoup plus que les plus gros dictionnaires ou encyclopédies jamais édités sur papier ! C'est le résultat d'un travail initié notamment par l'Université de Lausanne (merci à nos amis suisses) et poursuivi en France au CNRS et différentes universités dont Paris Descartes.

Christophe Pallier est un éminent chercheur en sciences cognitives, et je ne résiste pas à la tentation de reprendre ici la présentation qu'il fait de lui-même, les nuls en anglais me (lui) pardonneront :

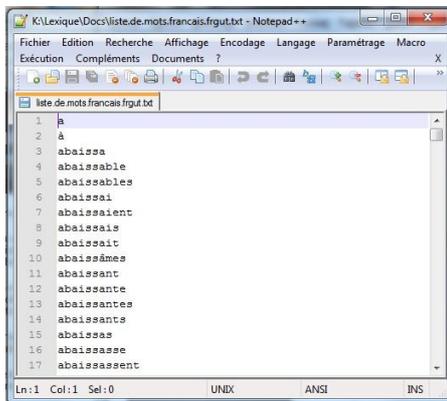
I am a middle age homo-sapiens drinking too much coffee. I am also the father of 3 homo-sapiens-sapiens (2 males/1 female). During the day, I work as a scientific researcher, sponsored by the french tax-payers to try and understand the brain machinery that allows us to speak. As no one seems willing to write a Wikipedia entry about me or my accomplishments, I have decided to create this website (in 1994, long before the advent of Wikipedia, but let us not be sidetracked by this apparent paradox).

Tout le monde a compris qu'il buvait trop de café, qu'il avait trois enfants, qu'il vivait aux crochets du contribuable et que Wikipedia ne parlait pas de lui !

Merci à ce brillant chercheur pour son humour et... l'autorisation qu'il nous a donnée d'utiliser sa publication (Il a exploité, souligne-t-il, les travaux de Christophe Pythoud et de l'association Gutenberg).

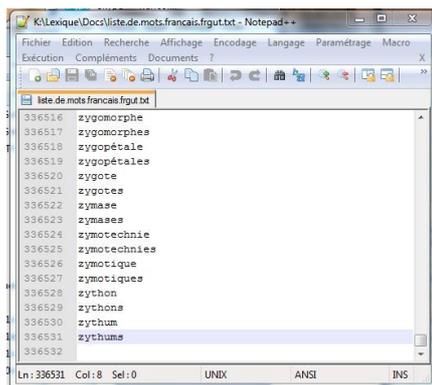
1.5. Lecture directe

Il suffit de cliquer sur le nouveau fichier pour que le programme par défaut de votre ordinateur affiche le texte. Le **bloc-notes** de Windows est lent et présente tous les mots sans retour à la ligne ; **WordPad** convient, mais **Notepad++** est mieux adapté, car les numéros de ligne sont indiqués :



Peut-on vérifier le nombre exact d'entrées ?

Il suffit de rejoindre la fin du fichier dans la fenêtre du programme. Notepad++ donne les informations cherchées :



Le dernier mot s'écrit **zythums** (Bière de l'Égypte ancienne) et il figure sur la ligne **336531**. Objectif atteint.

1.6. Utilisation

L'ensemble du fichier texte et de son programme de lecture donne accès aux fonctions essentielles :

- lecture d'un mot ;
- modification ;
- ajout ;
- recherche ;
- sauvegarde, etc.

Donc notre projet pourrait s'arrêter là, avec une belle collection de mots d'un côté, et un outil tout prêt pour la consulter...

Mais faisons un rêve. Ne pourrait-on pas :

- dessiner une interface plus accueillante ;
- disposer d'une petite fenêtre pour dialoguer avec la machine ;
- attacher une information, pas forcément une documentation complète, style wiki, mais au

moins un lien avec un autre mot...

C'est ce que l'on va aborder dans le prochain chapitre en utilisant Lazarus que l'on peut charger dès maintenant à partir du site Developpez.com ou directement ici : [Lien 09](#)

Attention, la page affiche des boutons Téléchargez ou Download qui sont des liens publicitaires.

La cible à repérer s'intitule

- Lazarus Windows 32 bits
- Lazarus Windows 64 bits
- ou Lazarus Mac, Linux, etc.

dans des versions datées du 28 ou 29 juillet 2012.

Cliquez sur la ligne qui vous convient et installez Lazarus.

1.7. Conclusion

Pour la langue française, la ressource disponible se révèle mieux que copieuse : notre projet démarre dans de bonnes conditions. Mais il est difficile d'imaginer que des ressources homologues n'existent pas pour d'autres langues : Internet est une mine féconde où il suffit souvent de chercher...

Sans oublier la colle du début de chapitre : les baignoires sont (également) des loges au théâtre ; les lavabos sont (également) des toilettes. Si on associe *théâtre* et *toilettes*, avec l'aide peut-être de quelques lettres déjà identifiées, on aboutit à... *entracte* !

Un essai avec Google : le moteur de recherche propose *Jacob Delafon*... Logique purement commerciale, loin de tout humour !

Voyons si nos petits bras musclés et nos quelques neurones encore valides seront en mesure de relever le défi...

Le prochain chapitre abordera différentes méthodes pour ouvrir le fichier texte avec Lazarus.

« Je ne retomberai jamais en enfance, j'y suis toujours resté. »

Tristan Bernard

2. Affichage sous Lazarus

2.1. Introduction

Le premier chapitre a conduit au chargement d'un fichier texte. Fichier quand même exceptionnel puisqu'il apporte plus de 336 000 mots de la langue française !

Cette liste est consultable à l'aide d'un programme courant, style bloc-notes ou WordPad, mais l'intérêt de cette consultation reste alors très restreint. Un outil tel que Lazarus va ouvrir des possibilités intéressantes. Dans ce chapitre seront présentées trois méthodes de lecture, et une technique d'évaluation.

Dans le chapitre suivant, le programme sera complété pour aborder différentes façons d'afficher et de balayer le texte.

2.2. Démarrage de Lazarus

Votre ordinateur dispose de Lazarus d'un côté, du fichier de mots d'un autre. Pour simplifier les explications à venir,

nous vous proposons de créer un répertoire **Lexique** dans lequel seront ouverts les sous-répertoires **Docs** et **Lex1**.

Le répertoire **Docs** recevra les documents textes et pour commencer, le fichier de mots.

Lex1 sera réservé à Lazarus, donc au programme source, au programme compilé et à l'ensemble des fichiers de service.

Pour les chapitres suivants, on créera le répertoire **Lex2**, **Lex3**, etc. de façon à faciliter à chaque instant le retour à un état antérieur du programme.

Lancez Lazarus, cliquez sur **Fichier/Nouveau/Application**. Au premier plan apparaît une fenêtre intitulée **Form1** : c'est elle qui recevra les boutons de commande et les modules d'affichage.

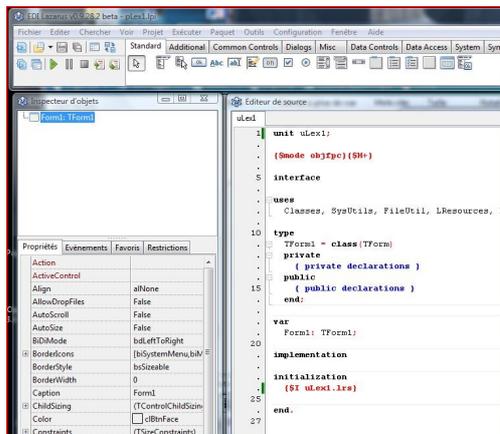
Au second plan s'ouvre la fenêtre de l'éditeur de code intitulée **Éditeur de source**.

2.3. Sauvegarde de l'unité

La première ligne de l'éditeur s'intitule `unit Unit1` ; cliquez sur **Fichier/Enregistrer sous**. Choisissez :

- votre répertoire de travail, par exemple **Lex1** ;
- le nom de votre unité, par exemple **uLex1**, sauvegardez, option **Préserver le nom**.

Le fichier **uLex1.pas** est créé dans le répertoire **Lex1**.



2.4. Sauvegarde du projet

Dans le menu, sous le mot **Fichier**, se présente une icône avec le menu contextuel **Nouvelle unité**. Sous cette icône, une autre avec l'information **Afficher les Unités**. Cliquez sur cette seconde icône : deux unités sont listées. La première, **uLex1**, est celle que l'on vient de créer. Cliquez sur la seconde et enregistrez sous le nom de **pLex1**.

L'unité et le projet sont en sécurité sur votre disque dur. Toutes les modifications seront enregistrées et Lazarus permettra un retour en arrière après toute une série de saisies.

Cette phase préalable d'enregistrement (choix du répertoire, enregistrement de l'unité et du projet) est fondamentale sous Lazarus, qui connaît dorénavant son espace de travail : votre programme est maintenant

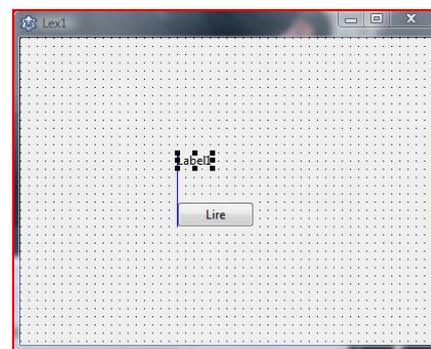
localisé et la compilation (construction du fichier exe) se fera dans le répertoire que vous aurez choisi.

2.5. Interface graphique

Pour l'instant, l'interface graphique se présente sous la forme d'une fenêtre intitulée **Form1**. Nous nous limiterons, dans l'immédiat, à trois modifications :

- **Nom de l'interface**. Dans le menu Fenêtre de Lazarus, sélectionnez **Form1**. Cliquez dans cette fenêtre pour indiquer à Lazarus que vous vous intéressez à cet objet précis. Effectivement, Lazarus, dans sa fenêtre **Inspecteur d'objets** a sélectionné la ligne **Form1:TForm1**. Au-dessous, dans l'onglet **Propriétés**, ligne **Caption**, on lit **Form1**. Remplacez ce nom par **Lex1** : immédiatement le nouveau nom s'affiche en tête de notre interface graphique ;

- **Bouton**.



Sous la ligne du menu, cliquez sur le bouton marqué **Ok**, puis cliquez au centre de la fenêtre **Lex1** : un bouton appelé **Button1** apparaît. Dans l'inspecteur d'objets, la sélection a basculé sur ce nouvel objet. Dans l'onglet **Propriétés**, ligne **Caption**, remplacez **Button1** par **Lire** ;

- **Texte**. Sous la ligne du menu, à côté du bouton, figure un objet qui se présente sous le sigle **Abc**. Cliquez dessus, puis cliquez dans l'interface graphique : l'objet **Label1** apparaît. Il est sélectionné automatiquement dans l'inspecteur d'objets.

L'interface est minimaliste, mais nous aurons l'occasion de l'enrichir !

2.6. Code

Nous voilà maintenant au pied du mur : nous imaginons un clic sur le bouton, qui déclenche la lecture du fichier texte, et une information apparaît sur l'interface pour signaler que l'opération s'est déroulée normalement.

Voici le code qui permet la lecture du fichier texte :

```
procedure LireListeDeMots;
var
  MotCour: string;
  lettre: char;
  fLex: file;
  listeMots : TStringList;
begin
  TopChro := 0;
  TopChrono('');
```

```

listeMots := TStringList.Create;
AssignFile(fLex,
'liste.de.mots.francais.frgut.txt');
  {$I-}
Reset(fLex, 1);
  {$I+}
if IOResult = 0 then
begin
  Seek(fLex, 0);
  MotCour := '';
  while not EOF(fLex) do
  begin
    BlockRead(fLex, lettre, 1);
    if lettre <> Chr(10) then
      MotCour := MotCour + lettre
    else
    begin
      listeMots.Append(MotCour);
      MotCour := '';
    end;
  end;
  CloseFile(fLex);
  TopChrono('Lecture octets');
  Form1.Label1.Caption := 'Dernier mot (n° '
+ IntToStr(listeMots.Count) + ') : ' +
listeMots.Strings[listeMots.Count-1];
end
else
  ShowMessage('Erreur d'accès au fichier');
Beep;
end;

```

Il faut copier ce texte dans l'éditeur de source, sous la ligne **implementation**. Le compilateur cherchera le fichier texte dans le répertoire de travail : il faut donc copier le fichier de mots (en principe dans le répertoire **Docs**) dans le répertoire de travail, **Lex1**.

Un clic sur le petit triangle vert (sous le mot **Éditer** dans le menu Lazarus) déclenche un contrôle général du projet, sa compilation et son exécution (en mode débogage).

Si rien ne se passe, c'est que tout va bien, le compilateur a fait son travail jusqu'au bout sans rencontrer de problème. Si une erreur est intervenue, lisez attentivement les informations présentées dans la fenêtre **Messages**. Corrigez selon les indications données.

Pour revenir en mode édition, il suffit de cliquer sur le carré rouge, à droite du triangle vert : le mode débogage est arrêté, l'édition du programme peut reprendre.

À quoi sert le bouton **Lire** ? Pour l'instant, le programme n'inclut aucune instruction pour le cas où l'on cliquerait dessus ! Cette omission va être rapidement corrigée.

Dans l'interface graphique **Lex1**, cliquez une fois sur le bouton **Lire** : son cadre est souligné par huit petits carrés permettant par exemple de modifier sa forme. Dans la fenêtre **Inspecteur d'objets**, l'objet **Button1** est alors sélectionné. Dans la partie inférieure de la fenêtre, dans l'onglet **Propriétés**, cliquez sur la ligne **OnClick** : à droite, trois points apparaissent ; cliquez dessus : l'éditeur de source est immédiatement enrichi d'une nouvelle procédure, et le curseur d'édition clignote entre les instructions begin et end ;

Une seule instruction est à écrire :

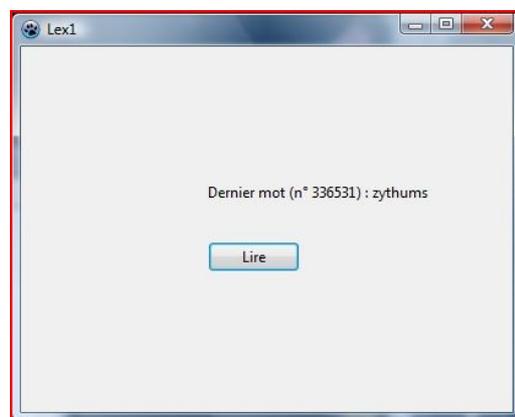
`LireListeDeMots`

qu'il faut terminer par un point-virgule « ; ».

Dans la pratique, notre programme **uLex1** fabrique (à l'aide de tout l'environnement Lazarus évidemment) une interface graphique **Lex1** avec un bouton et une ligne de texte.

Lorsque l'on clique sur le bouton, le programme exécute la procédure `LireListeDeMots`. Quand celle-ci est terminée, la ligne de texte nous présente les informations voulues.

Cliquez sur le petit triangle vert pour compiler. Cliquez sur le bouton **Lire** et... patientez, car la lecture du fichier de mots peut durer de 30 à 60 secondes selon le matériel utilisé. L'information voulue apparaît enfin :



2.7. Deuxième méthode

L'accès au contenu du fichier par l'intermédiaire de Lazarus semble bien lent. En effet, le code contient l'instruction

```
BlockRead(fLex, lettre, 1);
```

qui impose au matériel de lire le fichier lettre par lettre, chaque lettre occupant un octet. Comme le fichier a une longueur de 3,7 mégaoctets, il y a autant d'accès disque, ce qui est pénalisant.

La seconde méthode consiste à regrouper la lecture par blocs de textes (les mots) séparés par un octet de retour à la ligne. Le code devient alors :

```

procedure LireListe2;
var
  MotCour: string;
  fLex: TextFile;
  listeMots : TStringList;
begin
  listeMots := TStringList.Create;
  AssignFile(fLex,
'liste.de.mots.francais.frgut.txt');
  {$I-}
  Reset(fLex);
  {$I+}
  if IOResult = 0 then
  begin

```

```

while not EOF(fLex) do
begin
  ReadLn(fLex, MotCour);
  listeMots.Append(MotCour);
end;
CloseFile(fLex);
Form1.Label1.Caption := 'Lecture par mot ;
mot (n° '
  + IntToStr(listeMots.Count) + ') : ' +
listeMots.Strings[listeMots.Count-1];
end
else
  ShowMessage('Erreur d'accès au fichier');
Beep;
end;

```

Pour lancer cette procédure, on va ajouter un bouton, appelé **Lire2**, et dans la procédure **OnClick** de ce bouton, glisser l'instruction LireListe2 ; un clic sur le petit triangle vert pour compiler, un clic sur le bouton **Lire2** et... cette fois on apprécie une réaction presque immédiate.

Le premier mode de lecture imposait un déchiffrement caractère par caractère, et le second passe directement d'un mot à l'autre. En gros, les accès disques passent de 3 700 000 (nombre d'octets) à 330 000 (nombre de mots). Plus loin nous verrons comment évaluer précisément la performance.

2.8. Troisième méthode

Le rythme pourrait encore s'accélérer si la lecture pouvait s'opérer d'un bloc, avec une séparation automatique des mots...

Cela est possible en remplaçant les instructions BlockRead (version 1) ou ReadLn (version 2) par l'instruction de haut niveau

```
listeMots.LoadFromFile(nomFichier)
```

(listeMots est la liste utilisée précédemment). LoadFromFile indique tout simplement qu'il faut la remplir à partir du fichier précisé.

Le code se réduit encore :

```

procedure LireListeTexte;
var
  listeMots : TStringList;
begin
  listeMots := TStringList.Create;
  listeMots.LoadFromFile('liste.de.mots.francais.
  frgut.txt');
  Form1.Label1.Caption := 'Lecture 3 ; dernier
  mot (n° '
    + IntToStr(listeMots.Count) + ') : ' +
  listeMots.Strings[listeMots.Count-1];
  Beep;
end;

```

Pour lancer cette procédure, on va ajouter un bouton, appelé **Lire3**, et dans la procédure **OnClick** de ce bouton, glisser l'instruction :

```
LireListeTexte ;
```

Un clic sur le petit triangle vert pour compiler, un clic sur le bouton **Lire3** et... cette fois on apprécie encore. Simplicité et efficacité.

2.9. Chronomètre

Pour y voir clair sur les performances de tel ou tel programme, rien de tel que de mesurer la durée qui sépare le démarrage d'une tâche et son achèvement.

Il existe dans la bibliothèque LCL une fonction **GetTickCount** qui donne, en millièmes de seconde, la position du compteur Temps du processeur. Pour rendre cette fonction accessible, il suffit de rajouter dans la ligne uses l'unité **LclIntf** :

```

unit uLex1;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, FileUtil, LResources, Forms,
  Controls, Graphics, Dialogs,
  StdCtrls, LclIntf;

type

  { TForm1 }

  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Label1: TLabel;
    Memo1: TMemo;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;

```

Lazarus saura identifier l'instruction **GetTickCount** ; le code de la procédure pourrait s'écrire ainsi :

```

procedure TopChrono(Info : string);
begin
  if TopChro = 0 then
  begin
    TopChro := GetTickCount;
    Form1.Memo1.Append('Départ chrono');
  end
  else
    Form1.Memo1.Append(Info + ' Durée du
  processus '
    +
  IntToStr(GetTickCount-TopChro)+' ms');
end;

```

La procédure modifie la variable globale TopChro. Si elle vaut 0, le champ Memo signale le départ du chronomètre. Sinon, le Memo présente l'écart de temps, en millisecondes, depuis le départ.

Pour que ce code soit utilisable, il convient de :

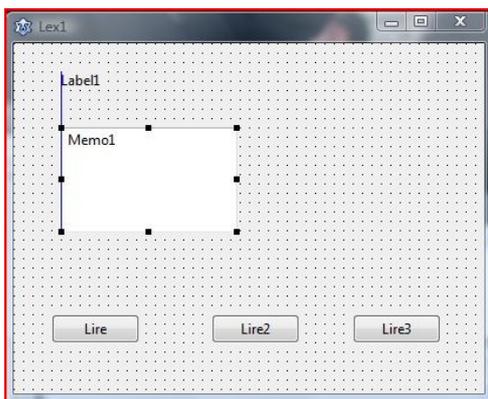
Primo, déclarer la variable globale TopChro ; pour cela, sous les lignes :

```
var Form : TForm1 ;
```

on ajoute :

```
TopChro : integer ;
```

Secundo, ajouter un composant **Memo** dans l'interface graphique : ce composant est présenté dans le menu Lazarus entre le bouton marqué **Ok**, déjà utilisé, et le bouton marqué **on**. On clique une fois sur le composant, une fois sur l'interface **Lex1** et on agence l'ensemble pour obtenir sensiblement la présentation suivante :



Le champ Memo dispose également de nombreuses propriétés directement accessibles dans l'inspecteur d'objets : cliquez sur la propriété Scrollbars ; faites passer le paramètre de `ssNone` à `ssAutoBoth`. Ainsi, les ascenseurs latéral et vertical permettront de naviguer sans difficulté dans ce composant.

Tertio, compléter chaque procédure de lecture par une mise à zéro de la variable globale et un premier appel à **TopChrono**, ceci en début de procédure, et un second appel à **TopChrono** en fin de procédure.

Pour vérifier que tout fonctionne, compilez, et cliquez successivement sur les boutons **Lire**, **Lire2** et **Lire3**.

Les mesures de durée sont affichées dans le mémo.

Sur une machine, on obtient par exemple :

| Type de lecture | Durée millisecondes |
|-----------------|---------------------|
| Octet par octet | 39171 |
| Mot par mot | 437 |
| Globale | 156 |

2.10. Conclusion

Une bonne utilisation du code permet de réaliser des économies spectaculaires dans les temps d'accès.

En l'occurrence, l'instruction `BlockRead` se révèle désastreuse par rapport à l'instruction `LoadFromFile`.

Alors, pourquoi l'avoir utilisée ?

D'abord pour les besoins de comparaison : aucune solution ne peut être *a priori* considérée comme la meilleure.

Ensuite parce que, la suite le montrera, cette instruction nous sera bien utile !

Le présent chapitre a montré comment Lazarus permet de lire un fichier de mots. Le suivant abordera l'affichage des mots et le balayage du fichier depuis le début jusqu'à la fin.

L'unité **uLex1** se présente dorénavant comme suit :

```
L'unité uLex1
unit uLex1;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, FileUtil, LResources, Forms,
  Controls, Graphics, Dialogs,
  StdCtrls, LclIntf;

type
  { TForm1 }

  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Label1: TLabel;
    Memo1: TMemo;
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure Button3Click(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;

var
  Form1: TForm1;
  TopChro : integer;
implementation

{ TForm1 }
procedure TopChrono(Info : string);
begin
  if TopChro = 0 then
  begin
    TopChro := GetTickCount;
    Form1.Memo1.Append('Départ chrono');
  end
  else
    Form1.Memo1.Append(Info + ' Durée du
processus '
                    +
IntToStr(GetTickCount-TopChro)+' ms');
end;

procedure LireListeDeMots;
var
```

```

MotCour: string;
lettre: char;
fLex: file;
listeMots : TStringList;
begin
  TopChro := 0;
  TopChrono('');
  listeMots := TStringList.Create;
  AssignFile(fLex,
'liste.de.mots.francais.frgut.txt');
  {$I-}
  Reset(fLex, 1);
  {$I+}
  if IOResult = 0 then
  begin
    Seek(fLex, 0);
    MotCour := '';
    while not EOF(fLex) do
    begin
      BlockRead(fLex, lettre, 1);
      if lettre <> Chr(10) then
        MotCour := MotCour + lettre
      else
      begin
        listeMots.Append(MotCour);
        MotCour := '';
      end;
    end;
    CloseFile(fLex);
    TopChrono('Lecture octets');
    Form1.Label1.Caption := 'Dernier mot (n° '
      + IntToStr(listeMots.Count) + ') : ' +
listeMots.Strings[listeMots.Count-1];
  end
  else
    ShowMessage('Erreur d'accès au fichier');
    Beep;
  end;

procedure LireListe2;
var
  MotCour: string;
  fLex: TextFile;
  listeMots : TStringList;
begin
  TopChro := 0;
  TopChrono('');
  listeMots := TStringList.Create;
  AssignFile(fLex,
'liste.de.mots.francais.frgut.txt');
  {$I-}
  Reset(fLex);
  {$I+}
  if IOResult = 0 then

```

```

begin
  while not EOF(fLex) do
  begin
    ReadLn(fLex, MotCour);
    listeMots.Append(MotCour);
  end;
  CloseFile(fLex);
  TopChrono('Lecture mots');
  Form1.Label1.Caption := 'Lecture par mot ;
mot (n° '
      + IntToStr(listeMots.Count) + ') : ' +
listeMots.Strings[listeMots.Count-1];
  end
  else
    ShowMessage('Erreur d'accès au fichier');
    Beep;
  end;

procedure LireListeTexte;
var
  listeMots : TStringList;
begin
  TopChro := 0;
  TopChrono('');
  listeMots := TStringList.Create;
  listeMots.LoadFromFile('liste.de.mots.francais.
frgut.txt');
  TopChrono('Lecture texte');
  Form1.Label1.Caption := 'Lecture 3 ; dernier
mot (n° '
      + IntToStr(listeMots.Count) + ') : ' +
listeMots.Strings[listeMots.Count-1];
  Beep;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
  LireListeDeMots;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  LireListe2;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  LireListeTexte;
end;

initialization
  {$I uLex1.lrs}

end.

```

Retrouvez l'article de dimanche2003 en ligne : [Lien 10](#)

SAS Batch : Utilisation du mode Checkpoint/Restart

L'article présente le mode checkpoint/restart permettant à un programme batch SAS interrompu par une erreur, de reprendre son exécution là où il s'était arrêté.

1. Introduction

L'utilisation du mode CheckPoint/Restart permet de soumettre un programme batch qui serait tombé en erreur et ce, à partir de la dernière étape DATA ou PROC valide.

Lorsqu'il est soumis de nouveau, le programme saute jusqu'à l'étape qui était en erreur et ne soumet pas les étapes précédentes.

L'objectif est de laisser la possibilité à l'utilisateur de corriger son programme et de pouvoir le soumettre de nouveau en évitant de tout exécuter depuis le début.

2. Modes de fonctionnement

Il existe deux modes, le premier permet de tracer les traitements et de marquer les erreurs ; le second permet de reprendre les traitements là où était marquée la première erreur (si celle-ci a été correctement corrigée).

2.1. Le mode Checkpoint

Grâce à ce mode, SAS enregistre des points de repère à chaque étape du programme dans le journal et dans un catalogue nommé CHECKPNT.SAS7BCAT qui contient les informations sur les étapes DATA et les PROC uniquement.

Un certain nombre de précautions doivent être prises en compte avec ce mode pour que tout fonctionne correctement.

Tout d'abord, le programme qui s'exécute doit accéder aux tables intermédiaires du traitement (notamment celles avant l'erreur) sinon il sera en incapacité de reprendre là où il en était. Par conséquent, si la WORK était utilisée, elle ne doit pas être supprimée en fin de session ou recrée à la prochaine. Pour cela, il faut utiliser les options NOWORKTERM et NOWORKINIT (si toutes les données sont enregistrées dans des bibliothèques dites permanentes, ces options ne sont pas utiles).

Ensuite, il est préférable de forcer SAS à s'arrêter dès la première erreur. Ceci est possible avec l'option ERRORCHECK STRICT.

Dans le cadre de l'utilisation de ce mode, voici la liste des options importantes :

- SYSIN : permet l'appel d'un programme SAS ;
- STEPCHKPT : active le mode CheckPoint ;
- STEPCHKPTLIB : permet d'indiquer le LIBREF dans lequel SAS doit sauver le catalogue. Il faut spécifier la commande LIBNAME en amont du

programme pour que SAS puisse l'affecter ;

- NOWORKTERM : permet de conserver la WORK à la fin de la session ;
- NOWORKINIT : évite d'initialiser une nouvelle WORK au démarrage de la session ;
- ERRORCHECK STRICT : force l'analyse syntaxique des étapes LIBNAME, FILENAME, %INCLUDE et LOCK ;
- ERRORABEND : force l'arrêt sur n'importe quelle erreur.

2.1.1. Illustration de la mise en place du mode checkpoint

Nous créons un appel de SAS sous forme de ligne de commande. Le programme exécuté se nomme « production » et nous utilisons les options précitées qui servent à contrôler le niveau d'erreur (avec ERRORCHECK STRICT, ERRORABEND) et d'activer le mode Checkpoint (avec STEPCHKPT).

Ici STEPCHKPTLIB permet de sauvegarder le catalogue dans le LIBREF savCheck. Par contre la WORK est détruite en fin de session :

```
Sas -sysin "c:\projet_alpha\production.sas"
-stepchkpt -stepchkptlib savCheck -errorcheck
strict -errorabend
```

Ici la WORK est conservée (avec NOWORKTERM NOWORKINIT). Nous verrons que ces deux options sont fondamentales à ce mode de travail.

```
sas -sysin "c:\projet_alpha\production.sas"
-stepchkpt -noworkterm -noworkinit -errorcheck
strict -errorabend
```

2.2. Le mode Restart

Il faut exécuter le programme avec l'option STEPSTART pour que le programme puisse être relancé à partir de la dernière erreur après que le catalogue CHECKPNT.SAS7BCAT ait été analysé. Dans ce cas, SAS relit le catalogue afin d'isoler les étapes s'étant correctement terminées et reprendre à celle qui était tombée en erreur.

Ce mode exécute tout d'abord les étapes générales telles que les LIBNAME, FILENAME et les compilations de macros car le catalogue contient uniquement les informations sur les étapes en erreur ou en succès. De cette façon, les éléments cruciaux ne sont pas omis.

Afin de contrôler un peu plus le mode Restart, il est

possible d'ajouter une option forçant SAS à exécuter une étape importante. Ce peut être une étape DATA ou la définition de macro-variables par exemple. Ceci se fait avec l'option CHECKPOINT EXECUTE_ALWAYS qui se place immédiatement avant l'étape considérée.

Enfin, le lancement du mode Restart peut également s'accompagner du lancement du mode Checkpoint. Comme le programme peut encore tomber en erreur, il sera possible de corriger et de relancer.

Ainsi nous aurons la ligne suivante qui cumule les deux modes :

```
sas -sysin "c:\projet_alpha\production.sas"
-stepchkpt -stepRestart -stepchkptlib savCheck
-errorcheck strict -errorabend
```

3. Exemples de mise en œuvre

Les exemples suivants vont illustrer l'utilisation du mode Checkpoint/Restart avec l'exécution d'un programme SAS sur une plate-forme Windows SAS 9.3. Ce procédé fonctionnera en 9.2 et sur UNIX.

3.1. Programme avec conservation de la WORK

Nous créons un fichier .bat permettant d'appeler le programme datametric.sas grâce à la commande SYSIN. Nous employons les options permettant de conserver la LOG et la WORK en plus du mode CheckPoint.

Le programme contient trois étapes simples : une étape DATA, une proc SORT et une étape DATA qui conclut le programme.

```
libname libtest "C:\Data\test";
%let choix=1;
data premiere;
  length rowid 8; * unique id;
  do level_1 = 0 to 2;
    do level_2 = 0 to 2;
      do x = 1 to 6+10*ranuni(1234);
        rowid + 1;
        y = sin(rowid/2);
        output;
      end;
    end;
  end;
run;
proc sort data=premiere;
by level_1;
run;
data second;
set premiere;
where level_1=&choix;
run;
```

Voici le contenu du fichier .bat qui appelle ce programme :

```
"C:\Program
Files\SASHome\SASFoundation\9.3\sas.exe"
-CONFIG "C:\Program
Files\SASHome\SASFoundation\9.3\nls\en\sasv9.cfg"
-LOG "C:\Data\test\monfichier.log"
-sysin "C:\Data\test\Datametric.sas"
-stepchkpt -noworkterm -noworkinit -errorcheck
strict -errorabend
```

Le journal du traitement qui suit montre clairement que le mode Checkpoint est activé avec l'insertion des balises CHECKPOINT avant chaque étape.

```
NOTE: Begin CHECKPOINT execution mode.
NOTE: Checkpoint library: c:\sasfiles\saswork\datametric.
NOTE: WORK library: c:\sasfiles\saswork\datametric.
NOTE: CHECKPOINT 1.
5      data premiere;
6          length rowid 8; * unique id;
7          do level_1 = 0 to 2;
8              do level_2 = 0 to 2;
9                  do x = 1 to 6+10*ranuni(1234);
10                     rowid + 1;
11                     y = sin(rowid/2);
12                     output;
13                 end;
14             end;
15         end;
16     run;
NOTE: The data set WORK.PREMIERE has 66 observations and 5 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.03 seconds
17
NOTE: CHECKPOINT 2.
18     proc sort data=premiere;
19     by level_1;
```

Pour constater l'arrêt sur une erreur, nous insérons une maladresse typographique dans le programme.

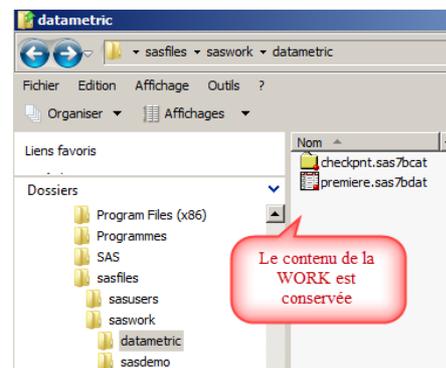
Le code suivant :

```
proc sort data=premiere;
by level_1;
run;
```

Devient :

```
proc sort data=premiere;
by level_1;
run;
```

Lorsque le programme tombe en erreur, la WORK est conservée avec le catalogue permettant à SAS de savoir où repartir ainsi que la première table générée (elle n'avait pas d'erreurs).



Nous relançons le programme corrigé avec l'option -STEPRESTART cette fois. Par précaution, nous laissons le mode Checkpoint afin de pouvoir stopper sur une prochaine erreur.

Contenu du fichier .bat :

```
"C:\Program
Files\SASHome\SASFoundation\9.3\sas.exe"
-CONFIG "C:\Program
Files\SASHome\SASFoundation\9.3\nls\en\sasv9.cfg"
-LOG "C:\Data\test\monfichier.log"
-sysin "C:\Data\test\Datametric.sas"
-stepchkpt -stepRestart -noworkterm -noworkinit
-errorcheck strict -errorabend
```

Cette fois, la première étape est reconnue comme étant sans erreur et donc est ignorée.

```
NOTE: Begin CHECKPOINT execution mode.
NOTE: Checkpoint library: C:\sasfiles\saswork\datametric.
NOTE: WORK library: C:\sasfiles\saswork\datametric.
NOTE: Begin CHECKPOINT-RESTART(2) execution mode.
NOTE: Checkpoint library: C:\sasfiles\saswork\datametric.
1 data preniere;
2 length rowid 8; * unique id;
3 do level_1 = 0 to 2;
4 do level_2 = 0 to 2;
5 do x = 1 to 6+10*ranuni(1234);
6 rowid + 1;
7 y = sin(rowid/2);
8 output;
9 end;
10 end;
11 end;
12 run;
13
NOTE: End CHECKPOINT-RESTART(2) draining and resume normal execution.
```

Démarrage du mode Checkpoint

L'étape est sautée car elle n'était pas erreur

Dans le journal, les étapes sont encore marquées en mode Checkpoint car l'option STEPCHKPT est utilisée par crainte d'une nouvelle erreur.

```
NOTE: End CHECKPOINT-RESTART(2) draining and resume normal execution.
NOTE: CHECKPOINT 2.
14 proc sort data=premiere;
15 by level_1;
16 run;
NOTE: There were 66 observations read from the data set WORK.PREMIERE.
NOTE: The data set WORK.PREMIERE has 66 observations and 5 variables.
NOTE: PROCEDURE SORT used (total process time):
real time 0.07 seconds
cpu time 0.03 seconds
17
18 %let choix=1;
19
NOTE: CHECKPOINT 3.
20 data second;
21 set premiere;
22 where level_1=&choix;
23 run;
NOTE: There were 20 observations read from the data set WORK.PREMIERE.
```

Arrivée à la dernière étape

3.2. Programme avec utilisation d'une WORK temporaire

Cet exemple prouve que même si le catalogue est sauvegardé dans un répertoire permanent alors que les données sont stockées dans une véritable WORK, alors le mode RESTART se trouve dans l'incapacité de rejouer le traitement.

Voici le contenu du nouveau .bat où les options de conservation de la WORK sont supprimées.

```
"C:\Program
Files\SASHome\SASFoundation\9.3\sas.exe"
-CONFIG "C:\Program
Files\SASHome\SASFoundation\9.3\nls\en\sasv9.cfg"
-LOG "C:\Data\test\monfichier.log"
-sysin "C:\Data\test\Datametric.sas"
-stepchkpt -STEPCHKPTLIB libtest -errorcheck
strict -errorabend
```

Si l'on indique l'option -STEPCHKPTLIB libtest dans la ligne de commande, le catalogue est conservé dans le répertoire lié au LIBREF libtest défini dans le programme (ici C:\Data\test).

Nous obtenons dans le journal les informations suivantes, puis la même erreur un peu plus bas.

```
NOTE: Begin CHECKPOINT execution mode.
NOTE: Checkpoint library: C:\Data\test.
NOTE: WORK library:
C:\sasfiles\saswork\datametric\_TD4132_WIN-
EH2M0MSIPC9_.
```

Après correction, nous relançons.

```
NOTE: Checkpoint library: C:\Data\test.
5 data preniere;
6 length rowid 8; * unique id;
7 do level_1 = 0 to 2;
8 do level_2 = 0 to 2;
9 do x = 1 to 6+10*ranuni(1234);
10 rowid + 1;
11 y = sin(rowid/2);
12 output;
13 end;
14 end;
15 end;
16 run;
NOTE: End CHECKPOINT-RESTART(2) draining and resume normal execution.
NOTE: CHECKPOINT 2.
18 proc sort data=premiere;
ERROR: File WORK.PREMIERE.DATA does not exist.
19 by level_1;
20 run;
NOTE: The SAS system stopped processing this step.
NOTE: SAS set option OBS=0 and will continue to check observations.
This might cause NOTE: No observations in data set.
```

Marquée sans erreur, l'étape est sautée

La table n'a pas été sauvegardée dans une bibliothèque permanente, elle n'existe plus.

Puisque la WORK n'était pas conservée et que les données n'étaient pas enregistrées dans un répertoire permanent, le programme tombe de nouveau en erreur car la première étape créant la table PREMIERE, n'a pas été rejouée : la table source de la seconde étape n'existe pas et retourne une erreur. C'est un cercle vicieux qui s'amorce car le programme sait d'où repartir mais n'a plus les données pour le faire.

Par conséquent, il faut conserver le catalogue et les données dans un répertoire qui n'est pas temporaire.

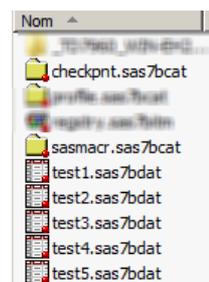
3.3. Utilisation du mode Checkpoint/Restart avec une boucle macro

On teste si le mode de marquage SAS rentre dans une boucle macro.

Ici le programme crée dix tables à partir d'une seule. Nous introduisons une erreur de syntaxe (un point-virgule manquant) qui créera une erreur uniquement sur l'étape n° 5.

```
%macro loop;
%do i= 1 %to 10;
data test&i;
set sashelp.class;
%if &i=5 %then %do;
i=2
%end;
run;
%end;
%mend;
%loop;
```

Comme dans le premier exemple, la ligne de commande exécutant ce programme va permettre de conserver la WORK et le catalogue qui sera dedans.



Le mode Checkpoint reconnaît parfaitement chaque étape DATA et stoppe sur l'erreur.

```
NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: The data set WORK.TEST3 has 19 observations and 5 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time             0.00 seconds

NOTE: CHECKPOINT 4.

NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: The data set WORK.TEST4 has 19 observations and 5 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time             0.00 seconds

NOTE: CHECKPOINT 5.

NOTE: Line generated by the invoked macro "LOOP".
14          run;
           22

ERROR 22-322: Syntax error, expecting one of the following: !, !!, &, *,
<=, <>,
=, >, ><, >=, AND, EQ, GE, GT, IN, LE, LT, MAX, MIN, NE, N
A=, |,
      |, ~=.

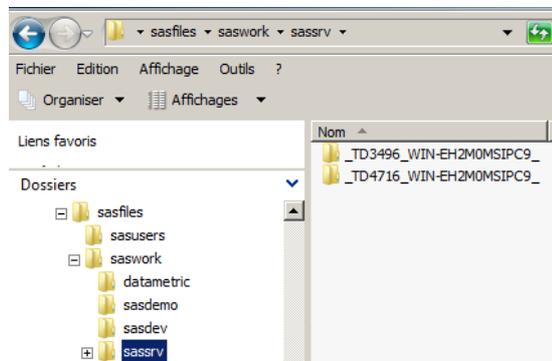
NOTE: The SAS System stopped processing this step because of errors.
NOTE: SAS set option OBS=0 and will continue to check statements.
      This might cause NOTE: No observations in data set.
WARNING: The data set WORK.TEST5 may be incomplete.  When this step was
0
      observations and 7 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time             0.01 seconds

ERROR: SAS ended due to errors.
      You specified: OPTIONS ERRORABEND;.
ERROR: Errors printed on page 2.
13
24, 2012                                The SAS System                                16:11
```

L'erreur de point-virgule stoppe le programme à la cinquième étape

```
-SASUSER "C:\sasfiles\users\!username\sasuser"
-WORK "C:\sasfiles\saswork\!username"
-UTILLOC "C:\sasfiles\utilloc"
```

Ce code signifie qu'une WORK est créée dans le sous-répertoire reprenant le nom de l'utilisateur. Ce code permet donc d'obtenir un résultat comme celui-ci pour l'utilisateur nommé sassrv :



Après correction, nous lançons la ligne de commande avec l'option STEPRESTART et nous constatons que le programme reprend à la cinquième étape et poursuit son travail.

```
NOTE: SAS initialization used:
      real time           0.22 seconds
      cpu time             0.26 seconds

1          %macro loop;
2
3          %do i= 1 %to 10;
4              data test&i;
5                  set sashelp.class;
6                  %if &i=5 %then %do;
7                      %f &i=5 %then %do;
8                          %end;
9                      %end;
10             %run;
11         %end;
12     %mend;
13 %loop;
14

NOTE: Begin CHECKPOINT execution mode.
NOTE: Checkpoint library: C:\sasfiles\saswork\datametric.
NOTE: WORK library: C:\sasfiles\saswork\datametric.

NOTE: Begin CHECKPOINT-RESTART(5) execution mode.
NOTE: Checkpoint library: C:\sasfiles\saswork\datametric.

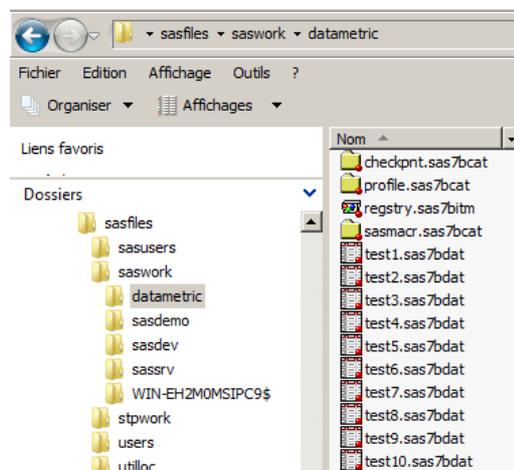
NOTE: End CHECKPOINT-RESTART(5) draining and resume normal execution.

NOTE: CHECKPOINT 5.

NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: The data set WORK.TEST5 has 19 observations and 6 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time             0.03 seconds

12
24, 2012                                The SAS System                                16
```

Lorsque l'option NOWORKTERM est utilisée, SAS stocke les tables directement dans le répertoire nommé dans le fichier sasv9.cfg sans créer de répertoire commençant par _TD_. Il n'en créera d'ailleurs pas d'autres, tout se déroulera dans ce seul répertoire. Pour le voir, nous lançons des programmes avec nos deux options sous le user Datametric. Nous constatons dans la capture suivante que tout est écrit directement dans C:\sasfiles\saswork\datametric :



4. Conservation des WORK

À ce stade il faut connaître le fonctionnement des options NOWORKTERM et NOWORKINIT pour comprendre comment les utiliser.

Ces deux options permettent de conserver la WORK et de ne pas en recréer une afin d'exploiter celle que la précédente session a créée. Dans ce cas, SAS n'agit pas comme à l'accoutumée en créant un répertoire dynamiquement mais utilise directement le répertoire spécifié dans l'option -WORK du fichier CFG (on peut concevoir que la WORK devient un répertoire unique et permanent).

Illustrons cela.

L'environnement suivant crée les WORK dans le répertoire suivant : C:\sasfiles\saswork\!username car le fichier sasv9.cfg contient une définition permettant de créer un répertoire sous le nom de l'utilisateur.

En conclusion, si ces options doivent être utilisées, il faut prendre garde à créer un répertoire WORK pour chaque programme lancé en batch afin que tous n'écrivent pas au même endroit. Il suffit pour cela d'ajouter la commande -WORK dans la ligne de commande SAS appelée dans le batch. Il faut également utiliser NOWORKINIT pour que SAS ne vide pas le contenu de la WORK à la prochaine ouverture de session.

5. Conclusion

Le mode Checkpoint/Restart permet de relancer des traitements à partir de l'étape tombée en erreur. Les deux modes se cumulant, il est possible de relancer un programme après l'avoir corrigé et de marquer de nouvelles erreurs. Le premier identifie l'étape, le second permet de repartir de l'étape incriminée.

Ce mode de fonctionnement est finalement assez pratique pour éviter de relancer tous les programmes d'un batch et leurs étapes.

Cependant, il faut prendre garde à utiliser le bon ensemble d'options pour que tout ceci fonctionne :

- il faut que toutes les tables soient accessibles afin que le programme puisse accéder aux données nécessaires pour repartir dans de bonnes conditions ;
- il faut conserver le catalogue dans une bibliothèque permanente pour que SAS le retrouve. Les options NOWORKTERM NOWORKINIT doivent donc être utilisées si l'on ne souhaite travailler en WORK plutôt que dans

des bibliothèques permanentes ;

- si des étapes importantes doivent être obligatoirement exécutées, l'étape CHECKPOINT EXECUTE_ALWAYS doit être placée avant elles.

Les administrateurs peuvent adopter aisément une stratégie de prise en charge des programmes en production avec ces deux modes. Une même ligne de commande acceptant les options STEPCHKPT et STEPSTART, il n'est pas utile de créer plusieurs .bat selon que l'on soit en mode Checkpoint ou Restart. Il faut noter enfin que tout n'est pas automatisable car une intervention humaine reste nécessaire pour corriger le programme.

Retrouvez l'article de DATAMETRIC en ligne : [Lien 11](#)

Comment classer les données dans des tables liées et construire un formulaire père/fils - Pourquoi et comment implanter un sous-formulaire avec ACCESS

Dans ce tutoriel, je vous propose une marche à suivre pour les cas où l'on veut présenter sur un écran une série d'informations à deux niveaux, par exemple :

- chaque facture avec la date, le destinataire, les conditions de paiement... au 1er étage et le détail des articles facturés juste en dessous ;
- un produit fini et la liste des pièces détachées qui le composent ;
- ou comme ce qui va nous servir de modèle dans ce tutoriel, un plat cuisiné et la liste des ingrédients nécessaires.

1. Prérequis

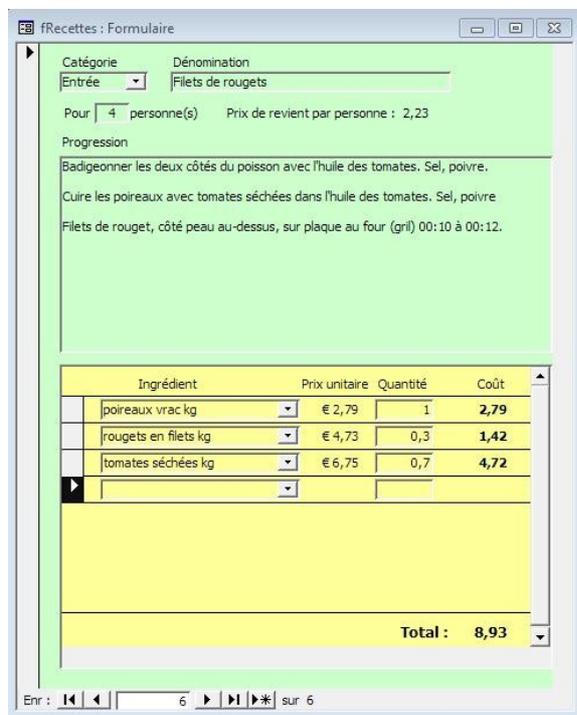
Certaines connaissances rudimentaires de base sont censées acquises : créer une table, construire une requête, un formulaire...

Ce texte s'adresse à des utilisateurs Access débutants.

2. Notre objectif

Construire un formulaire principal présenté en mode « simple » (un seul enregistrement affiché à la fois) et un sous-formulaire en mode continu (une liste d'autant de lignes que d'enregistrements) pour donner le détail.

Comme ceci :



Pour plus de lisibilité, le fond du formulaire principal est en vert. Celui du sous-formulaire est en jaune.

Ce formulaire est volontairement « minimaliste », j'aurais pu y ajouter :

- une image pour illustrer chaque plat ;

- l'accès en un clic à une vidéo logée sur Internet ;
- permettre des recherches multicritères : selon la catégorie, le nom du plat, un ingrédient...

J'ai voulu me focaliser sur « le » sujet et aussi éviter tout recours à la programmation : il n'y a pas une seule ligne de code VBA dans ce tutoriel.

Je me propose de traiter ces thèmes dans un tutoriel à venir. Vous y reviendrez, si ça vous intéresse, lorsque vous ne serez plus débutant... Mais si c'est urgent, voyez deux tutoriels de référence :

cafeine : Recherche multicritère ([Lien 12](#)) ;

Jeannot45 : Recherche multicritère avec ou sans code ([Lien 13](#)).

3. Organiser les données utiles à l'application

Pour la rédaction de ce chapitre, je me suis largement inspiré de l'article Office : « Concepts de base sur la conception d'une base de données » ([Lien 14](#)).

En vrac, les données utiles sont : la catégorie (entrée, plat principal...), la dénomination de la préparation, le texte de la recette, les quantités de chaque ingrédient et leur prix, le prix de revient par convive.

3.1. Classement « tout en une »

Dans un premier temps, nous pourrions imaginer une table dans laquelle on stocke toutes les informations nécessaires :

| Catégorie | Dénomination | Progression | Ingrédient1 | PU1 | Quant1 | Coût1 | Ingrédient2 | PU2 | Quant2 | Coût2 | ... | NbreConvives | PRConvive |
|-----------|------------------|-----------------------------|----------------------|------|--------|-------|--------------------|------|--------|-------|-----|--------------|-----------|
| Dessert | Tiramisu | Casser 4 œufs et séparer | Amarettto versé | 0,85 | 1 | 0,85 | boudoirs ps | 1,64 | 1 | 1,64 | | 5 | 1,07 |
| Dessert | Gâteau Malakof | Préparer 1/2 litre de crème | boudoirs ps | 1,64 | 1 | 1,64 | saï 1 litre | 0,74 | 0,5 | 0,37 | | 4 | 1,45 |
| Entrée | Filets de rouget | Badigeonner les deux | rougets en filets kg | 4,73 | 0,3 | 1,42 | tomates séchées kg | 6,75 | 0,3 | 2,03 | | 4 | 1,56 |

Une telle solution présente plusieurs inconvénients :

- il faut chaque fois répéter des données identiques : dénomination, le nom des ingrédients, leur prix ;
- tenir cette table à jour sera fastidieux : à chaque variation d'un prix unitaire, il faudra modifier à plusieurs endroits... sans en oublier ! ;
- combien de quadruplets Ingrédient/PU/Quantité/Coût faut-il prévoir ?

D'une recette à l'autre, le nombre d'ingrédients varie : pour la recette d'un œuf dur, un quadruplet suffit, pour un plat plus élaboré, il en faudra une douzaine ! Et sans doute, plus tard le chef trouvera une recette pour laquelle il manque des cases : la loi de Murphy s'applique aussi en cuisine...

Dans cette approche, chaque ligne de la table contient des informations de plusieurs sujets :

- relatifs au service : entrée, plat principal ou dessert ;
- relatifs à la recette : son nom, la progression, le nombre de convives... ;
- relatifs à chaque ingrédient : son nom, son prix unitaire...

3.2. Classement « par sujet »

Dans notre exemple, nous avons trois sujets d'informations :

- les **catégories** : parle-t-on d'une entrée, d'un plat principal, d'un dessert ? ;
- les **plats** : comment les nomme-t-on ? Comment les prépare-t-on ? Pour combien de convives ? ;
- les **ingrédients** : comment les nomme-t-on ? Quel est leur prix ?

Nous constituons ces trois tables avec chacune une colonne du type NuméroAuto qui leur servira de clé primaire (un élément qui permet d'identifier de manière unique un enregistrement dans la table).

Les voici :

tblCategories : Table

| idCategorie | Categorie |
|-------------|-----------|
| 1 | Entrée |
| 2 | Principal |
| 3 | Dessert |

tblPlats : Table

| idPlat | Denomination | idCategorie | Progression | NbreConvives |
|--------|-----------------------|-------------|-------------------------------|--------------|
| 1 | Tiramisù | 3 | Casser 4 œufs et séparer les | 5 |
| 2 | Filets de soles aux f | 2 | Voir vidéo http://www.youtube | 1 |
| 3 | Régime | 2 | Sans commentaire | 1 |
| 4 | Gâteau Malakov | 3 | Préparer ½ litre de crème pât | 4 |
| 5 | Soupe à l'oignon | 1 | Voir vidéo http://www.youtube | 4 |
| 6 | Filets de rougets | 1 | Badigeonner les deux côtés c | 4 |

tblIngredients : Table

| idIngrédient | Ingrédient | Prix |
|--------------|----------------------------|--------|
| 1 | œuf pc. | € 0,25 |
| 2 | sucré semoule ultra fin kg | € 1,39 |
| 3 | cassonade kg | € 2,04 |
| 4 | mascarpone kg | € 6,94 |
| 5 | sucré vanillé sachet | € 0,10 |
| 6 | boudoirs pq | € 1,64 |
| 7 | Amaretto verre | € 0,85 |

4. Établir des relations entre les tables

Nous devons maintenant trouver un moyen de rassembler les sujets de façon cohérente.

Dans le formulaire que nous voulons construire :

Recettes : Formulaire

Catégorie: Dénomination: Filets de rougets

Entrée

Pour 4 personnes(s) Prix de revient par personne : 2,23

Progression

Badigeonner les deux côtés du poisson avec l'huile des tomates. Sel, poivre.

Cuire les poireaux avec tomates séchées dans l'huile des tomates. Sel, poivre

Filets de rouget, côté peau au-dessus, sur plaque au four (gril) 00:10 à 00:12.

| Ingrédient | Prix unitaire | Quantité | Coût |
|----------------------|---------------|----------|------|
| poireaux vrac kg | € 2,79 | 1 | 2,79 |
| rougets en filets kg | € 4,72 | 0,3 | 1,42 |
| tomates séchées kg | € 6,75 | 0,7 | 4,72 |

Total : 8,93

- la donnée 1 provient de la table *tblCategories* ;
- la donnée 2 provient de la table *tblPlats* ;
- les données 3 proviennent de la table *tblIngredients* ;
- les données 4 sont le résultat d'une opération. En principe, on ne stocke pas ce genre d'information dans une table si on dispose déjà des éléments qui permettront à Access d'effectuer le calcul en temps opportun.

4.1. Relation un-à-plusieurs

Si nous considérons *tblPlats*, nous constatons qu'à chaque ligne correspond une (et une seule) ligne dans la table *tblCategories* : chaque plat repris dans *tblPlats* est soit une entrée, un plat principal ou un dessert.

De manière symétrique, à chaque ligne de *tblCategories* correspondent plusieurs lignes de *tblPlats*.

Ce type de relation est appelé une relation « un-à-plusieurs ».

Pour matérialiser cette correspondance, nous allons ajouter une colonne dans *tblPlats* (le côté **plusieurs**) pour y mentionner la valeur de la clé primaire correspondante de *tblCategories* (le côté **un**).

tblPlats : Table

| idPlat | Denomination | idCategorie | Progression | NbreConvives |
|--------|-----------------------|-------------|-------------------------------|--------------|
| 1 | Tiramisù | 3 | Casser 4 œufs et séparer les | 5 |
| 2 | Filets de soles aux f | 2 | Voir vidéo http://www.youtube | 1 |
| 3 | Régime | 2 | Sans commentaire | 1 |
| 4 | Gâteau Malakov | 3 | Préparer ½ litre de crème pât | 4 |
| 5 | Soupe à l'oignon | 1 | Voir vidéo http://www.youtube | 4 |
| 6 | Filets de rougets | 1 | Badigeonner les deux côtés c | 4 |

Nous avons baptisé cette colonne dans *tblPlats* du même nom qu'elle porte dans *tblCategories*. Ce n'est pas obligatoire.

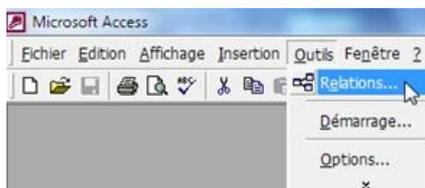
Par contre, les deux types de donnée doivent être identiques (un champ « NuméroAuto » est un « Entier long »).

Access pourra utiliser *idCategorie* de *tblPlats* pour trouver le nom de la catégorie correspondant à chaque plat.

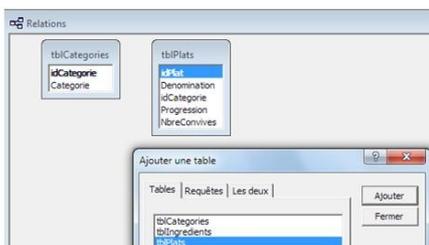
La colonne *idCategorie* dans *tblPlats* est appelée une **clé étrangère**. Une clé étrangère est la clé primaire d'une autre table.

Voici la suite des opérations pour communiquer cette relation à Access :

- dans la barre des menus, cliquez sur « Outils » et ensuite sur « Relations... » :

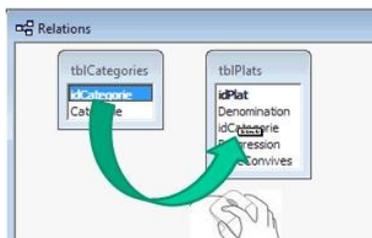


- une fenêtre s'ouvre et vous invite à sélectionner les tables, en l'occurrence *tblCategories* et *tblPlats* :

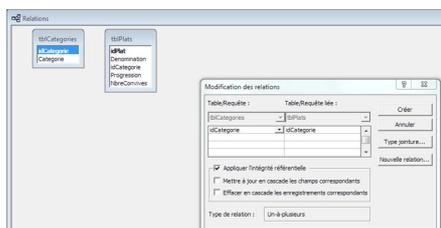


vous fermez la fenêtre « Ajouter une table » ;

- vous cliquez sur la clé primaire (*idCategorie* dans *tblCategories*) et en maintenant la pression, vous faites glisser sur la clé étrangère (*idCategorie* dans *tblPlats*) :



- une nouvelle fenêtre s'ouvre :



et vous cochez « Appliquer l'intégrité référentielle ». Cela

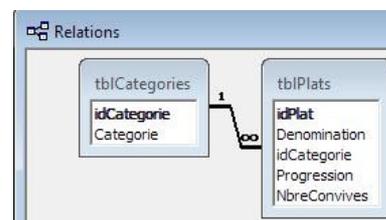
pour garantir la cohérence entre les enregistrements des tables liées : il sera impossible d'ajouter un enregistrement dans la table du côté « plusieurs » de la relation si la liaison ne peut s'établir avec une valeur de la clé de la table côté « un » de la relation.

Dans *tCategories* (côté « un » de la relation) **idCategorie** peut prendre les valeurs 1, 2 ou 3.

Conséquence de l'intégrité référentielle, dans *tPlats* (côté « plusieurs » de la relation), aucun enregistrement ne pourra être ajouté avec un **idCategorie** différent de 1, 2 ou 3.

De même, dans *tCategories* on ne pourra supprimer un enregistrement tant que des enregistrements existent dans *tPlats* avec cette valeur de **idCategorie** ;

- vous cliquez sur le bouton « Créer » et Access affiche la liaison comme ceci :

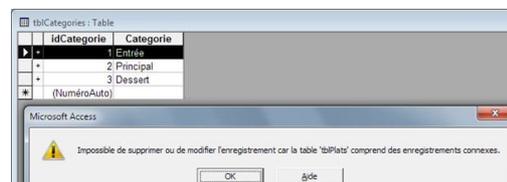


Pour expérimenter ce que signifie « Intégrité référentielle », essayez ceci :

- ouvrez la table *tblCategories* ;
- vous constatez une colonne supplémentaire ;
- cliquez sur un « + », Access vous affiche les enregistrements de *tblPlats* qui y sont liés :

| tblCategories : Table | | | | |
|-----------------------|--------------|----------------|-------------------------------|--------------|
| | idCategorie | Categorie | | |
| + | 1 | Entrée | | |
| + | 2 | Principal | | |
| - | 3 | Dessert | | |
| | idPlat | Denomination | Progression | NbreConvives |
| | 1 | Tiramisù | Casser 4 oeufs et séparer les | 5 |
| | 4 | Gâteau Malakov | Préparer ½ litre de crème pât | 4 |
| * | oAuto | | | 0 |
| * | (NuméroAuto) | | | |

- essayez de supprimer l'enregistrement 1, Access refuse :



- pressez la touche <ESC> pour annuler ;
- ouvrez *tblPlats* et dans la colonne *idCategorie*, essayez d'introduire « 4 », Access refuse :



- pressez la touche <ESC> pour annuler.

4.2. Une relation « plusieurs-à-plusieurs »

Considérons maintenant *tblPlats* et *tblIngredients*.

Dans la plupart des cas, un plat comprendra plusieurs ingrédients et un ingrédient interviendra probablement dans plus d'un plat.

Nous sommes devant une relation « plusieurs-à-plusieurs ».

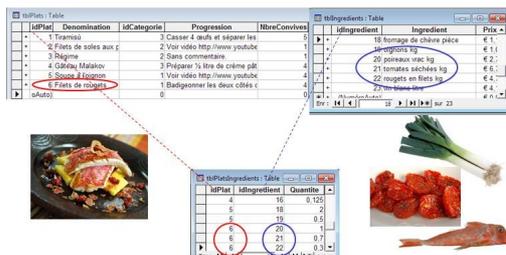
Cette fois, il ne suffit pas d'ajouter une clé étrangère dans *tblPlats* pour établir la liaison avec *tblIngredients*. Pour avoir plusieurs ingrédients dans un plat, vous devriez avoir plusieurs enregistrements par plat dans la table *tblPlats*, chacun avec l'idIngredient correspondant.

Et le même souci si on aborde par l'autre bout : si nous ajoutons la clé étrangère idPlat dans *tblIngredients*, il nous faudrait alors répéter les informations de l'ingrédient pour chaque plat différent.

4.3. Une table de jointure pour scinder une relation plusieurs-à-plusieurs en deux relations un-à-plusieurs

La solution : créer une table supplémentaire, appelée « **table de jointure** », qui va substituer **deux** relations un-à-plusieurs à la relation plusieurs-à-plusieurs.

L'astuce consiste à ajouter la clé primaire de chacune des deux tables dans la nouvelle table et d'y stocker chaque occurrence de relation.



Dans la foulée, c'est dans cette *tblPlatsIngredients* que nous logerons l'information des quantités.

Réflexion sur le choix de la clé primaire

La clé primaire doit permettre d'identifier sans ambiguïté une ligne quelconque de la table.

Il faut donc garantir que sa valeur soit différente d'une ligne à l'autre.

Pour la table *tblIngredients*, nous aurions pu choisir le nom de l'ingrédient en clair comme clé primaire : « oignons kg », « boudoirs pq »... tous les ingrédients ont un nom différent.

Alors, pourquoi avoir choisi un NuméroAuto comme clé ? Parce que l'expérience montre que c'est plus pratique !

Cette clé primaire sert souvent de clé étrangère dans une

autre table, c'est le cas dans notre exemple : l'information « boudoirs pq » est aussi incluse dans *tblPlatsIngredients* (il en faut pour le tiramisu et pour le Malakof).

Imaginez qu'au lieu de la valeur « 6 », nous ayons choisi la valeur « boudoirs pq » dans *tblPlatsIngredients* pour établir la relation. Cela fonctionnerait.

Mais si subitement, vous vouliez changer : au lieu de « boudoirs » vous vouliez « Madeira » ou plus générique « biscuits à la cuillère », non seulement il faudrait modifier dans *tblIngredients* mais aussi dans *tblPlatsIngredients*. Cela risque de vous demander des acrobaties : neutraliser temporairement l'intégrité référentielle, modifier partout sans rien oublier, rétablir l'intégrité...

Bref, comme clé primaire, il vaut mieux choisir une valeur qui ne risque pas d'évoluer dans le temps.

Choisir une valeur NuméroAuto comme clé primaire est une bonne garantie : Access attribue automatiquement une valeur à votre place. Un identificateur de ce type ne contient aucune information factuelle décrivant la ligne qu'il représente et vous ne serez jamais tenté de le modifier... vous n'y avez pas accès !

C'est une règle, évidemment, il y a des exceptions !

Dans *tblPlatsIngredients*, si nous considérons l'ensemble des valeurs des deux colonnes : IdPlat et IdIngredient, nous avons une combinaison unique qui ne risque pas de changer. Nous pouvons donc choisir cette combinaison comme clé primaire de notre table de liaison.

De plus, cela garantira qu'un même ingrédient n'est renseigné qu'une seule fois dans la composition d'une recette.

Lorsqu'une clé primaire se compose de plusieurs colonnes, on parle de **clé composite**.

Pour définir une clé composite :

- vous affichez la table en mode création ;
- vous sélectionnez les champs concernés et dans la

barre des menus, vous cliquez sur l'icône

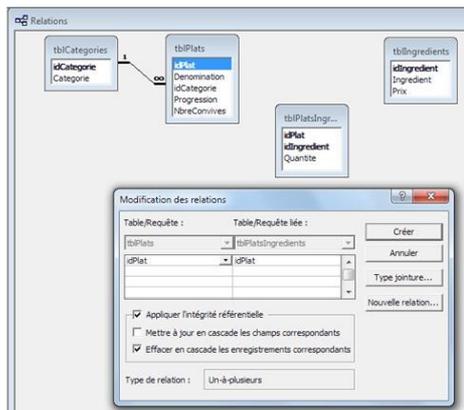
| tblPlatsIngredients : Table | |
|-----------------------------|-----------------|
| Nom du champ | Type de données |
| idPlat | Numérique |
| idIngredient | Numérique |
| Quantité | Numérique |

4.4. Compléter le modèle

Pour afficher les relations : dans la barre des menus, cliquez sur Outils/Relations...

Cliquez sur le bouton et ajoutez les tables *tblPlatsIngredients* et *tblIngredients*.

Faites glisser l'idPlat de *tblPlats* sur celui de *tblPlatsIngredients* et dans la fenêtre « Modification des relations », cochez « Appliquer l'intégrité référentielle » et « Effacer en cascade les enregistrements correspondants » :



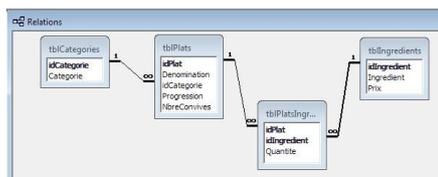
« Effacer en cascade les enregistrements correspondants », cela signifie que si l'utilisateur supprime un plat dans la table *tPlats*, tous les enregistrements de *tPlatsIngredients* qui portent l'idPlat en question seront automatiquement supprimés.

Logique ! Si nous décidons d'éliminer une recette de notre base de données, nous n'avons plus besoin de connaître les ingrédients qui permettent de la réaliser.

Si nous avons coché « Mettre à jour en cascade les champs correspondants », cela aurait voulu dire que si nous modifions une valeur d'idPlat dans *tPlats*, par exemple en remplaçant le idPlat « 2 » par la valeur « 28 », alors automatiquement dans *tPlatsIngredients* tous les idPlat « 2 » étaient modifiés en « 28 ».

Une telle fonctionnalité n'est pas utile dans notre cas : idPlat dans *tPlats* est du type NuméroAuto, il ne peut donc pas être modifié par l'utilisateur.

Procédons de même avec l'idIngredient de *tbiIngredients* et celui de *tbiPlatsIngredients*, le graphique des relations devient :



5. Première ébauche du formulaire fRecettes

Si vous voulez apprendre de bonnes pratiques pour la construction d'un formulaire, voyez ce tutoriel de Jean-Philippe AMBROSINO (argyronet) : Lien [15](#). Concentrez-vous sur le chapitre « 2. Présentation du formulaire exemple ». (Il sera encore temps, lorsque vous aurez acquis davantage d'expérience, de découvrir les autres chapitres de son tutoriel.) Prenez dès le début de bonnes habitudes de nommage. Si vous aimez la rigueur, voyez les Conventions typographiques du même auteur : Lien [16](#).

Passons en revue les différents contrôles de ce formulaire.

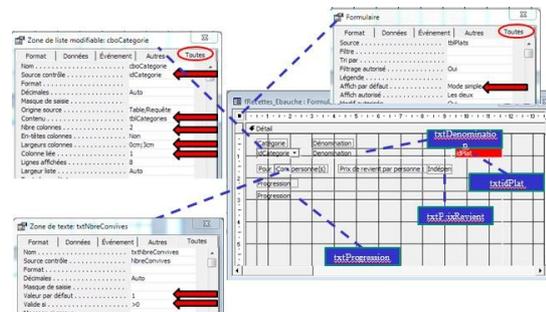
Pour examiner les propriétés, affichez le formulaire en mode « Création » et double-cliquez sur le carré supérieur gauche :



Cliquez ensuite sur un contrôle pour afficher ses propriétés.

Pour vous documenter sur une propriété de la liste, cliquez-la (elle se met en surbrillance et pressez la touche <F1> : l'aide Access s'ouvre alors à la bonne page.

5.1. Vue d'ensemble

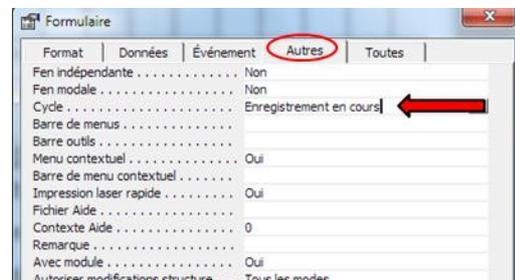


5.2. Propriétés du formulaire fRecettes

Sa source est la table *tPlats*.

Il affichera les enregistrements de la table en « mode simple ».

De plus, nous fixons la propriété Cycle à « Enregistrement en cours » pour éviter le passage intempestif à l'enregistrement suivant lorsqu'on navigue dans les contrôles.



5.3. Propriétés de cboCategorie

Cette zone de liste modifiable sera alimentée (propriété « contenu ») par la table *tCategories*.

Dans « Nbre de colonnes », nous exprimons qu'il faut considérer les deux premières colonnes de la table (elle n'en contient d'ailleurs que deux !).

Dans « Largeurs de colonnes », nous définissons l'espace réservé à l'affichage des colonnes lors du déploiement de la liste. Dans ce cas particulier : 0 cm pour la 1^{re} colonne signifie que nous voulons l'occulter.

Quand l'utilisateur aura choisi une ligne de la table, c'est la valeur de la première colonne à largeur non nulle qui sera affichée. Dans notre exemple, la deuxième colonne.

Par contre dans « Colonne liée », nous exprimons que la

valeur du contrôle sera celle de la 1^{re} colonne.

Pour vous en assurer, faites ce test :

- ouvrez **fRecettes** en mode « formulaire », le contrôle affiche « Dessert », cliquez sur le champ pour le rendre actif ;
- ouvrez la fenêtre d'exécution (<Ctrl> + G) et saisissez :

```
? forms!fRecettes!cboCategorie.value
```

et <Entrée> ;

De ce côté de la barrière, Access s'exprime dans la langue de Shakespeare ou plutôt de Jean-Claude Van Damme. Traduction en français : montrer (?) le contenu du contrôle **cboCategorie** actuellement affiché dans le formulaire **fRecettes** parmi la collection des formulaires (**forms**) actuellement actifs dans la base de données.

- Access vous affiche « 3 » ;
- saisissez :

```
? forms!fRecettes!cboCategorie.text
```

- Access affiche « Dessert ».

Remarquez également la propriété « Limiter à liste » égale à « Oui » : l'utilisateur sera obligé de choisir l'une des trois lignes proposées lorsque la zone de liste modifiable est dépliée.

Si on veut pouvoir choisir « Entremets », il faudra d'abord modifier la table *tCategories*...

5.4. Propriétés de txtNbreConvives

La propriété « Source » est NbreConvives.

La propriété « Valeur par défaut » est 1.

La propriété « Valide si » est > 0 donc obligatoirement un nombre positif.

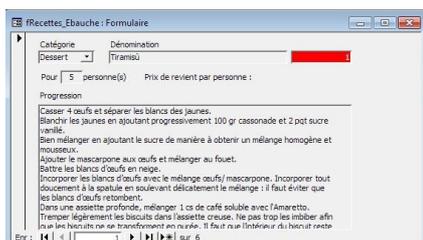
5.5. Propriétés de txtPrixRevient

Nous reviendrons plus tard sur la propriété « Source » de ce contrôle.

5.6. Propriétés de txtIdPlat

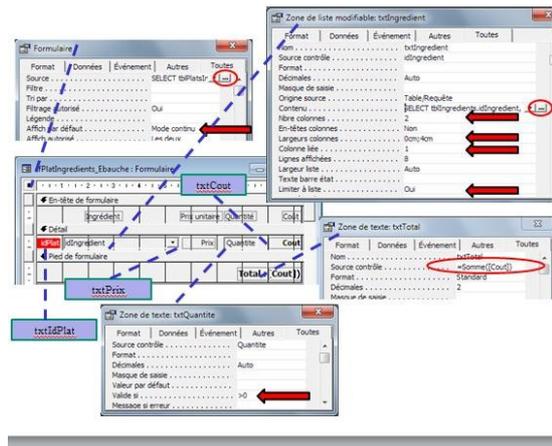
C'est le contrôle en rouge. Sa source est idPlat. Nous reviendrons plus tard sur l'utilité de ce contrôle.

5.7. L'aspect de fRecettes à ce stade



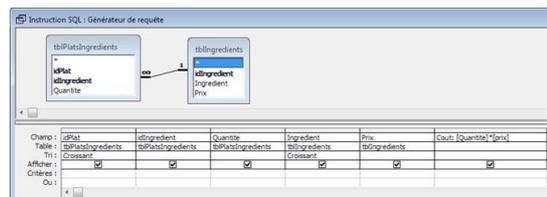
6. Le formulaire sfPlatIngredients

6.1. Vue d'ensemble



6.2. Propriétés du formulaire sfPlatIngredients

Sa source est une requête dont on peut voir le graphique en cliquant sur la propriété et sur les **...** qui apparaissent alors au bout de la ligne. La voici :



Nous considérons donc les tables liées *tblIngredients* et *tblPlatsIngredients*.

Sur la ligne Champ :

- nous prenons dans notre requête toutes les colonnes de *tblPlatsIngredients* ;
- nous ajoutons les colonnes « Ingredient » et « Prix » de *tblIngredients* ;
- nous construisons une colonne « Cout » en multipliant le champ Quantité par le Prix (remarquez les crochets qui encadrent le nom des colonnes).

Sur la ligne Tri :

- d'abord sur idPlat ;
- ensuite sur Ingredient (en clair).

Vue :

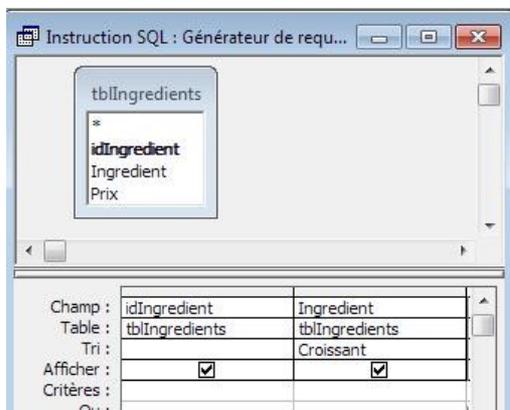
| idPlat | Quantite | IdIngredient | Ingredient | Prix | Cout |
|--------|----------|--------------|-------------------------|---------|--------------|
| 1 | 1 | 7 | Amaretto verre | € 0.85 | 0.85 |
| 1 | 1 | 6 | boudoirs pg | € 1.64 | 1.64 |
| 1 | 0.1 | 3 | caissonade kg | € 2.04 | 0.204000030 |
| 1 | 0.5 | 4 | mascarpone kg | € 6.94 | 3.47 |
| 1 | 4 | 1 | oeuf pc. | € 0.25 | 1 |
| 1 | 2 | 5 | sucres vanille sachet | € 0.10 | 0.2 |
| 2 | 0.15 | 8 | saies en filets kg | € 26.00 | 3.900000155 |
| 2 | 1 | 17 | un peu de tout | € 3.00 | 3 |
| 3 | 0.17 | 9 | pomme kg | € 1.80 | 0.306000032 |
| 4 | 0.125 | 12 | amandes pilées kg | € 2.90 | 0.3625 |
| 4 | 1 | 13 | ananas boîte | € 0.96 | 0.96 |
| 4 | 0.125 | 16 | beurre kg | € 6.50 | 0.8125 |
| 4 | 1 | 6 | boudoirs pg | € 1.64 | 1.64 |
| 4 | 0.05 | 15 | kirsch litre | € 20.50 | 1.0250000153 |
| 4 | 0.5 | 10 | lait litre | € 0.74 | 0.37 |
| 4 | 1 | 11 | poudre pudding sachet | € 0.37 | 0.37 |
| 4 | 0.125 | 14 | sucres impalpable kg | € 2.08 | 0.26 |
| 5 | 2 | 18 | fromage de chèvre pilée | € 1.19 | 2.38 |
| 5 | 0.5 | 19 | poignons kg | € 1.04 | 0.52 |
| 6 | 1 | 20 | poireaux vrac kg | € 2.79 | 2.79 |
| 6 | 0.3 | 22 | rougets en filets kg | € 4.73 | 1.4190000564 |
| 6 | 0.7 | 21 | tomates séchées kg | € 6.75 | 4.7249999195 |

6.3. Propriétés de txtIngredient

Sa source est idIngredient. Pourtant c'est l'ingrédient en « clair » qui sera affiché.

Voyez le contenu de la liste, c'est le SQL d'une requête :
SELECT [tblIngredients]...

Pour voir cette requête sous sa forme graphique, cliquez à l'endroit marqué , il vient :



Donc deux colonnes :

- la 1^{re}, celle qui est liée, contient l'idIngredient qui sera affecté à la source du contrôle ;
- c'est la 2^e qui sera affichée puisque les largeurs sont « 0cm;4cm ».

6.4. Propriétés de txtPrix, txtCout

Les sources sont respectivement : Prix et Cout. Les autres propriétés n'appellent pas de commentaire.

6.5. Propriétés de txtIdPlat

C'est le contrôle en rouge. Sa source est idPlat. Nous reviendrons plus tard sur l'utilité de ce contrôle.

6.6. Propriétés de txtQuantite

La propriété « Valide si » > 0 pour empêcher :

- l'oubli (la valeur par défaut est 0, puisque de type numérique) ;
- l'introduction d'une quantité négative.

6.7. Propriétés de txtTotal

La source est « =Somme([Cout]) », c'est-à-dire la somme de la colonne « Cout » de la requête.

N.B. [Cout] réfère à une colonne de la source du formulaire et non au contrôle (qui d'ailleurs s'appelle « txtCout »).

6.8. L'aspect de sfPlatIngredients à ce stade

| Ingrédient | Prix unitaire | Quantité | Coût |
|-------------------------|---------------|----------|------|
| Amaratto verre | € 0,85 | 1 | 0,85 |
| boudoirs pq | € 1,64 | 1 | 1,64 |
| cassonade kg | € 2,04 | 0,1 | 0,20 |
| mascarpone kg | € 6,94 | 0,5 | 3,47 |
| œuf pc. | € 0,25 | 4 | 1,00 |
| sucres vanillés sachet | € 0,10 | 2 | 0,20 |
| soles en filets kg | € 26,00 | 0,15 | 3,90 |
| un peu de tout | € 3,00 | 1 | 3,00 |
| poivre kg | € 1,80 | 0,17 | 0,31 |
| amandes pilées kg | € 2,90 | 0,125 | 0,36 |
| ananas boîte | € 0,96 | 1 | 0,96 |
| beurre kg | € 6,50 | 0,125 | 0,81 |
| boudoirs pq | € 1,64 | 1 | 1,64 |
| hirsch litre | € 20,50 | 0,05 | 1,03 |
| lait litre | € 0,74 | 0,5 | 0,37 |
| poudre pudding sachet | € 0,37 | 1 | 0,37 |
| sucres impalpables kg | € 2,08 | 0,125 | 0,26 |
| fromage de chèvre pièce | € 1,19 | 2 | 2,38 |
| poignons kg | € 1,04 | 0,5 | 0,52 |
| poireaux vrac kg | € 2,79 | 1 | 2,79 |
| fougets en filets kg | € 4,73 | 0,3 | 1,42 |
| tomates séchées kg | € 6,75 | 0,7 | 4,72 |

6.9. Un petit bonus

Essayons d'ajouter un ingrédient pour l'idPlat N° 6, par exemple 1 kg d'amande pilée :

| | | | |
|----------------------|--------|-----|------|
| fougets en filets kg | € 4,73 | 0,3 | 1,42 |
| tomates séchées kg | € 6,75 | 0,7 | 4,72 |
| amandes pilées Kg | € 2,90 | 1 | 2,90 |

La ligne se complète mais le total reste inchangé. C'est parce que l'enregistrement dans la table n'est pas encore réalisé. D'ailleurs le  en début de ligne est là pour vous le signaler.

Pour enregistrer vous pouvez soit :

- cliquer sur le sélecteur ;
- passer à un autre enregistrement du formulaire ;
- cliquer dans le formulaire ailleurs que sur la ligne courante ;
- passer à un autre plat ;
- dans la barre des menus : Enregistrement > Sauvegarder l'enregistrement ;
- utiliser le raccourci <Maj + Entrée>.

Si vous le faites, le total passe à 35,10 et  disparaît.

N'oubliez pas de supprimer cet ingrédient de votre recette : c'était juste pour voir comment Access réagit !

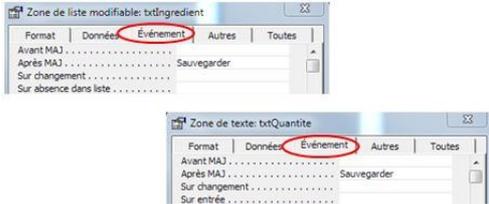
Le bonus que je vous propose, c'est de rendre cette sauvegarde automatique.

Nous allons créer une macro qui va émuler la pression des touches <Maj + Entrée> et déclencher l'exécution de cette macro après chaque mise à jour du contrôle « cboIngredient » ou du contrôle « txtQuantite ».

Voici la macro (nous l'avons appelée « Sauvegarder ») :



Et pour provoquer son déclenchement après chaque mise à jour de cboIngredient ou txtQuantité, nous modifions la propriété « Après MAJ » de ces deux contrôles :

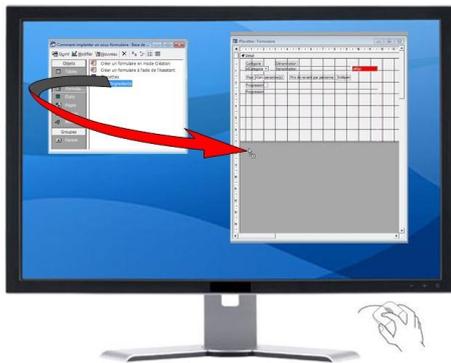


7. Incorporation de sfPlatIngredients dans fRecettes

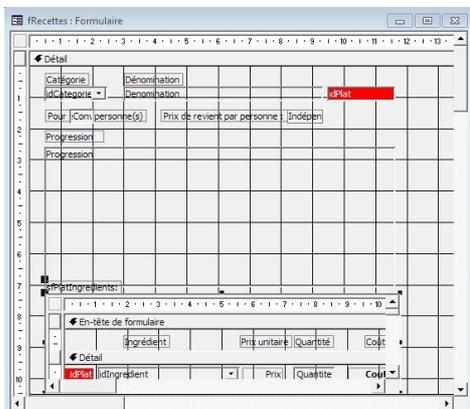
Nous allons intégrer *sfPlatIngredients* en tant que sous-formulaire de *fRecettes*.

Affichez à l'écran la fenêtre de la base de données à l'onglet « Formulaires » et *fRecettes* en mode « Création ».

Dans la fenêtre de la base de données, vous cliquez sur « sfPlatIngredients » en maintenant la pression sur le bouton de la souris et vous faites glisser-déposer dans *fRecettes* :



Quand vous relâchez la pression, *fRecettes* devient :



Access a intégré un nouveau contrôle dans *fRecettes*.

Affichez les propriétés de ce contrôle :



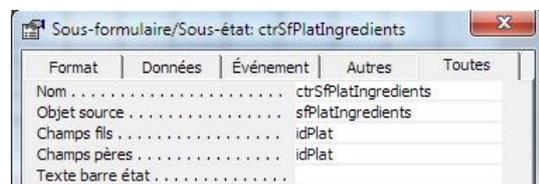
Constatez qu'il ne s'agit pas des propriétés de *sfPlatIngredients* !

Soyons précis : ce que nous avons fait, c'est ajouter à notre formulaire un **contrôle Sous-formulaire/Sous-état**, disons une espèce de container qui, dans notre exemple, pointera sur le formulaire *sfPlatIngredients*.

Lorsque vous sélectionnez ce contrôle, il affiche d'abord les propriétés du container.

Si ensuite vous cliquez sur l'un des contrôles du formulaire inclus, alors Access vous affiche les propriétés de ce contrôle comme si vous étiez dans le formulaire d'origine.

Pour plus de clarté, nous allons d'ailleurs modifier le nom qu'Access a attribué par défaut à ce contrôle. Nous l'appellerons « ctrSfPlatIngredients » :



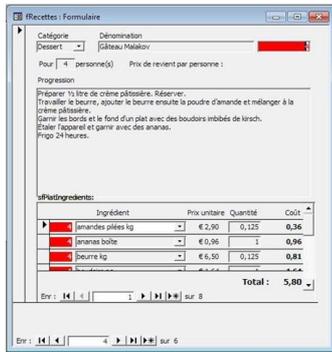
Passons en revue quelques propriétés du container « ctrSfPlatIngredients ».

Dans « Champs fils » et « Champs pères », on indique le nom des colonnes des tables ou requêtes qui servent de source au formulaire « père » et au formulaire « fils ». Cela va servir à coordonner l'affichage du père et du fils : le formulaire « fils » montrera seulement les enregistrements de sa source qui sont en relation avec l'enregistrement actuellement actif dans le formulaire « père ».

Remarquez que les champs fils et pères sont les noms des colonnes des tables et non pas le nom des contrôles (ils ne sont d'ailleurs pas encadrés de crochets et de plus les contrôles s'appellent : « txtIdPlat »).

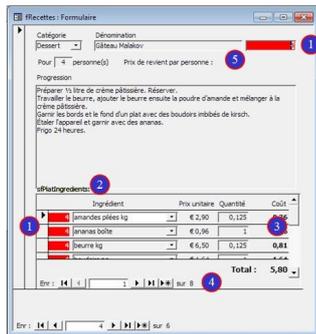
Dans notre exemple, ces noms sont identiques : ce n'est pas obligatoire, c'est la conséquence des nommages que nous avons choisis lors de la création des tables.

À ce stade voici comment se présente notre formulaire *fRecettes* :



Naviguez parmi les enregistrements : la liste des ingrédients est chaque fois limitée à ceux qui interviennent dans l'IdPlat affiché dans le formulaire principal (ici 4).

8. Quelques retouches



1 : dans les deux formulaires, nous avons un contrôle nommé txtIdPlat (ils s'affichent en rouge). Ils n'ont aucun intérêt pour l'utilisateur final. Je les avais prévus pour des raisons didactiques : pour montrer que la liste des ingrédients était limitée à ceux du plat.

Nous supprimons ces contrôles.

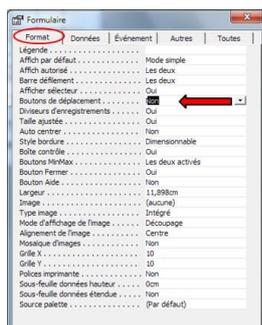
2 : nous supprimons l'étiquette, créée par défaut, du contrôle ctrSfPlatIngredients.

3 : nous agrandissons la hauteur du contrôle ctrSfPlatIngredients pour donner plus d'espace à l'affichage du sous-formulaire.

4 : il y a deux barres de boutons de déplacement :

- celle du formulaire principal : elle permet de se déplacer parmi les recettes ;
- celle du sous-formulaire : elle permet de se déplacer dans la liste des ingrédients.

Cette dernière n'est pas très utile, nous la supprimons.

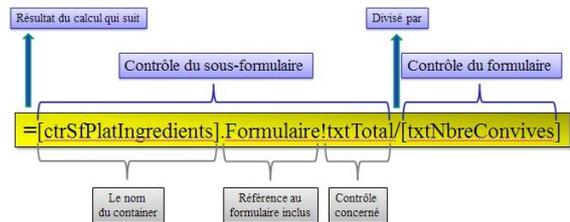


5 : et enfin la source du contrôle txtPrixRevient.

C'est le résultat de la division :

- du montant affiché dans le contrôle txtTotal du sous-formulaire
- par
- le montant affiché dans le contrôle txtNbreConvives du formulaire principal.

Voici l'explication de la syntaxe :



9. En guise de conclusion

Pour que votre apprentissage soit plus efficace, je vous suggère : de copier les tables dans une base de données vierge et de reconstruire le formulaire vous-même, en suivant le tutoriel pas à pas.

Autre bon tuyau

Pour se documenter sur les propriétés d'un formulaire ou d'un état, ou de leurs contrôles :

- afficher l'objet en mode construction ;
- cliquer sur la propriété : elle se met en surbrillance ;
- enfoncer la touche <F1> : l'aide Access s'ouvre à la bonne page.

On peut aussi :

- ouvrir l'aide <F1>, choisir l'onglet « Aide intuitive » et suivre les instructions ;
- ouvrir la fenêtre d'exécution (<Ctrl>+G), saisir un mot-clé, y placer le curseur de la souris et enfoncer <F1>.

La démarche proposée dans ce tutoriel vaut aussi pour des états et sous-états.

Pour vous familiariser, voyez aussi Implanter un sous-état dans un état de Jean BALLAT : Lien [17](#).

Vous pouvez télécharger ici la base de données Access2000 qui m'a servi à construire l'exemple : Lien [18](#).

Retrouvez l'article de Claude Leloup en ligne : [Lien 19](#).

Le programme de rentrée de la rubrique C++

Le mois de septembre est souvent associé à la fin des vacances et des belles journées ensoleillées, pour ceux qui ont eu la chance d'en avoir.

Pour faciliter la reprise du travail, la rubrique C++ de Developpez.com vous propose un remède simple : vous plonger dans la série d'articles passionnants, traduits par l'équipe de rédaction.

Les Guru of the Week ne sont plus à présenter ([Lien 20](#)). Ces articles de référence, couvrant de nombreux points du langage C++, restent incontournables pour les débutants et les autres. Cette série d'articles, commencée un peu avant l'été, va se poursuivre jusqu'à ce que tous les articles soient traduits.

Pour ceux qui sont intéressés par l'expressivité et les fonctionnalités haut niveau du C++, nous vous proposons les traductions des articles d'Eric Niebler sur Boost Proto ([Lien 21](#)) et les langages orientés domaine enfoui (DSEL en anglais, pour domain-specific embedded language). Nous vous proposons également des articles de John Carmack sur l'analyse statique de code et la

programmation fonctionnelle en C++ ([Lien 22](#)).

Ceux qui sont intéressés par la compréhension bas niveau du C++ ne seront pas oubliés. La série d'articles de Bruce Dawson aborde la représentation des nombres à virgule flottante en informatique et les problématiques liées à leurs manipulations ([Lien 23](#)). Les articles d'Alex Darby entrent plus en détail sur le code assembleur généré et comment éviter les pièges ([Lien 24](#)).

Comme d'habitude, chaque nouvelle publication sera annoncée sur le forum C++ ([Lien 25](#)) et le portail C++ ([Lien 26](#)).

Bonne rentrée à tous

Quels sont les articles qui vous intéressent le plus dans ce que nous vous proposons ?

Quels autres thèmes souhaiteriez-vous que l'on aborde ?

Si vous souhaitez participer aux traductions, vous pouvez contacter Guillaume Belz par message privé : [Lien 27](#).

Commentez la news de Guillaume Belz en ligne : [Lien 28](#).

Les derniers tutoriels et articles

Les nombres flottants et leurs pièges

La série d'articles de Bruce Dawson aborde en détail les problématiques liées à la représentation des nombres à virgule flottante. Ce premier article pose les bases et explore le monde étrange et merveilleux des mathématiques à virgule flottante.

1. Introduction

Il y a quelques années, j'ai écrit un article expliquant comment comparer des nombres à virgule flottante en utilisant des comparaisons d'entiers. Cet article a été plutôt populaire (il est fréquemment cité et les exemples de code ont été utilisés par nombre d'entreprises) et ça me fait un peu peur, parce que l'article a quelques défauts. Je ne vais pas donner de lien vers l'article : je veux le remplacer, et donc ne pas envoyer les gens le lire.

Aujourd'hui, je vais commencer par poser les bases en expliquant comment et pourquoi cette astuce fonctionne, tout en explorant le monde étrange et merveilleux des mathématiques à virgule flottante.

Il y a beaucoup de références qui expliquent la disposition binaire et le décodage des nombres à virgule flottante. Dans ce billet, je vais présenter l'organisation binaire et expliquer comment fonctionne le procédé de décodage par des expérimentations et de la rétroconception.

2. Norme IEE 754-1985

Le standard IEEE 754-1985 ([Lien 29](#)) spécifie le format des nombres à virgule flottante de 32 bits, type connu sous

le nom float dans de nombreux langages. La version 2008 du standard ([Lien 30](#)) ajoute de nouveaux formats, mais ne modifie pas les existants, qui sont standardisés depuis plus de 25 ans.

Un flottant (Note de traduction : par abus de langage, l'auteur utilise régulièrement le terme « flottant » à la place de « nombre à virgule flottante ») de 32 bits consiste en un champ d'un bit pour le signe, un champ de 8 bits pour l'exposant et un champ de 23 bits pour la mantisse. L'union ci-dessous montre la disposition des bits d'un tel flottant. Cette union nous sera très utile pour explorer et travailler sur la représentation interne des nombres à virgule flottante. Je ne recommanderais pas d'utiliser cette union pour du code de production (elle est en violation des règles d'aliasing (Note de traduction : on parle d'aliasing quand deux objets de deux types différents sont à la même adresse mémoire. Voir Wikipédia - Aliasing (computing) pour plus de détails : [Lien 31](#)) établies par certains compilateurs et ceux-ci vont donc probablement générer du code inefficace), mais elle nous sera utile pour apprendre. Ces articles sur l'aliasing ([Lien 32](#)) et l'undefined behavior (comportement indéfini) ([Lien 33](#)) proposent des détails supplémentaires.

```

union Float_t
{
    Float_t(float num = 0.0f) &#160;: f(num) {}
    // extraction portable des composants
    bool Negative() const { return (i >>
31) &#160; != 0; }
    int32_t RawMantissa() const { return i & ((1
<< 23) - 1); }
    int32_t RawExponent() const { return (i >>
23) & 0xFF; }

    int32_t i;
    float f;
#ifdef _DEBUG
    struct
    {
        // Champs de bits pour l'exploration. Ne
pas utiliser en production.
        uint32_t mantissa &#160;: 23;
        uint32_t exponent &#160;: 8;
        uint32_t sign &#160;: 1;
    } parts;
#endif
};

```

Le format des nombres flottants de 32 bits a été attentivement conçu pour leur permettre d'être réinterprété comme un entier et l'aliasing de i et f devrait fonctionner sur la majorité des plates-formes (si, comme gcc et VC++, elles autorisent l'aliasing par les unions), avec le bit de signe de l'entier et du flottant occupant la même position.

La disposition des champs de bits est dépendante du compilateur, donc la structure à champs de bits parts, qui est dans l'union, peut ne pas fonctionner sur toutes les plates-formes. Cependant, ceci fonctionne avec Visual C++ sur x86 et x64, ce qui est suffisant pour ce que je souhaite tester. Sur des systèmes big endian (Note de traduction : big endian se traduit par « gros-boutiste » et définit la représentation mémoire des nombres. Il dépend du processeur. Voir Wikipédia - Endianness pour plus de détails : Lien [34](#)) comme SPARC et PPC, l'ordre dans la structure à champs de bits est inversé.

3. Représentation de quelques nombres

Afin de vraiment comprendre les flottants, il est important d'explorer et d'expérimenter. Une façon d'explorer est d'écrire du code comme ceci, en compilant en mode « debug » afin que le débogueur ne le retire pas en l'optimisant :

```

void TestFunction()
{
    Float_t num(1.0f);
    num.i -= 1;
    printf("Valeur flottante, representation,
signe, exposant, mantisse\n");
    for (;;)
    {
        // point d'arrêt ici
        printf("%1.8e, 0x%08X, %d, %d, 0x%06X\n",
            num.f, num.i,
            num.parts.sign, num.parts.exponent,
            num.parts.mantissa);
    }
}

```

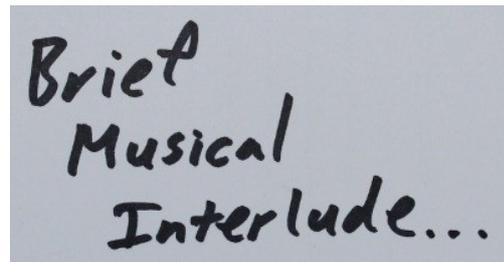
Placez un point d'arrêt sur la ligne du printf, puis ajoutez

les différents composants de num à votre fenêtre de débogueur et examinez-les, ce qui pourra donner cela par exemple :

| Name | Value | Type |
|--------------------|------------|--------------|
| num.f | 0.99999994 | float |
| num.i | 0x3f7ffff | int |
| num.parts.sign | 0x00000000 | unsigned int |
| num.parts.exponent | 0x0000007e | unsigned int |
| num.parts.mantissa | 0x007ffff | unsigned int |

Vous pouvez alors commencer à faire des tests interactifs, comme incrémenter la mantisse ou l'exposant, incrémenter num.i, ou inverser le bit de signe. En faisant cela, observez num.f afin de voir de quelle façon il évolue. Ou bien, assignez différentes valeurs de flottants à num.f et regardez comment les autres champs changent. Vous pouvez soit observer les résultats dans la fenêtre du débogueur, soit appuyer sur Run après chaque changement, afin que le printf s'exécute et affiche des résultats joliment formatés.

Allez-y. Placez Float_t et le code d'exemple dans un projet et jouez avec pendant quelques minutes. Découvrez les valeurs minimum et maximum des valeurs flottantes. Expérimentez avec les valeurs minimum et maximum de mantisse dans des combinaisons variées. Pensez aux conclusions. Ceci est la meilleure façon d'apprendre. J'attendrai.



J'ai placé quelques-uns des résultats que vous pouvez rencontrer pendant ces expériences dans le tableau ci-dessous :

| Valeur flottant | Représentation entière | Signe | Exposant | Mantisse |
|-----------------|------------------------|-------|----------|------------|
| 0.0 | 0x00000000 | 0 | 0 | 0 |
| 1.40129846e-45 | 0x00000001 | 0 | 0 | 1 |
| 1.17549435e-38 | 0x00800000 | 0 | 1 | 0 |
| 0.2 | 0x3E4CCCCD | 0 | 124 | 0x4CCCCD |
| 1.0 | 0x3F800000 | 0 | 127 | 0 |
| 1.5 | 0x3FC00000 | 0 | 127 | 0x400000 |
| 1.75 | 0x3FE00000 | 0 | 127 | 0x600000 |
| 1.99999988 | 0x3FFFFFFF | 0 | 127 | 0x7FFFFFFF |
| 2.0 | 0x40000000 | 0 | 128 | 0 |
| 16,777,215 | 0x4B7FFFFFFF | 0 | 150 | 0x7FFFFFFF |
| 3.40282347 | 0x7F7FFFFFFF | 0 | 254 | 0x7FFFFFFF |

| | | | | |
|----------------|------------|---|-----|---|
| e+38 | | | | |
| Infini positif | 0x7f800000 | 0 | 255 | 0 |

Avec ces informations nous pouvons commencer à comprendre le décodage des flottants. Les flottants utilisent un format en écriture scientifique base 2, donc nous pouvons nous attendre à ce que le décodage soit mantisse $\times 2^{\text{exposant}}$. Cependant, dans l'encodage de 1.0 et 2.0 la mantisse est nulle, donc comment ceci peut-il fonctionner ? Ça fonctionne grâce à une astuce. Les nombres normalisés en notation scientifique base 2 sont toujours de la forme :

$$1.xxxx \times 2^{\text{exposant}}$$

donc stocker le chiffre 1 initial est inutile. En l'omettant, on gagne un bit supplémentaire de précision – le champ de 23 bits d'un flottant gère en fait 24 bits de précision car il y a un 1 implicite avec une valeur de 0x800000.

L'exposant pour 1.0 devrait être nul mais le champ exposant vaut en fait 127. C'est parce que l'exposant est stocké sous une forme augmentée de 127. Pour convertir la valeur dans le champ exposant en la valeur de l'exposant, vous devez simplement soustraire 127.

Les deux exceptions à cette règle de calcul de l'exposant sont quand le champ exposant vaut 255 ou 0. 255 est une valeur pour exposant qui indique que le flottant est soit l'infini soit un NaN (« Not-a-Number »), une mantisse contenant zéro indiquant l'infini. 0 est une valeur spéciale pour exposant, qui indique qu'il n'y a pas de chiffre 1 initial sous-entendu, ce qui signifie que ces nombres ne sont pas normalisés. C'est nécessaire afin de représenter exactement la valeur zéro. La valeur de l'exposant dans ce cas vaut -126, ce qui est la même valeur que lorsque le champ exposant vaut 1.

Pour clarifier les règles d'exposant, j'ai ajouté une colonne « valeur de l'exposant » qui montre l'exposant binaire réel correspondant au champ exposant.

| Valeur flottante | Représentation entière | Signe | Champ exposant | Valeur de l'exposant | Mantisse |
|------------------|------------------------|-------|----------------|----------------------|----------|
| 0.0 | 0x00000000 | 0 | 0 | -126 | 0 |
| 1.40129846e-45 | 0x00000001 | 0 | 0 | -126 | 1 |
| 1.17549435e-38 | 0x00800000 | 0 | 1 | -126 | 0 |
| 0.2 | 0x3E4CCCD | 0 | 124 | -3 | 0x4CCCD |
| 1.0 | 0x3F800000 | 0 | 127 | 0 | 0 |
| 1.5 | 0x3FC00000 | 0 | 127 | 0 | 0x40000 |
| 1.75 | 0x3FE00000 | 0 | 127 | 0 | 0x60000 |
| 1.99999988 | 0x3FFFFF | 0 | 127 | 0 | 0x7FFFFF |
| 2.0 | 0x40000000 | 0 | 128 | 1 | 0 |

| | | | | | |
|----------------|-----------------|---|-----|---------|--------------|
| | 000 | | | | |
| 16,777,215 | 0x4B7FFF FFF | 0 | 150 | 23 | 0x7FFFF F |
| 3.40282347e+38 | 0x7F7FFF FFF | 0 | 254 | 127 | 0x7FFFF F |
| Infini positif | 0x7f8000 00 | 0 | 255 | Infini! | 0 |

Bien que ces exemples ne le montrent pas, les nombres négatifs sont gérés en mettant 1 dans le champ de signe, ce qui est appelé une forme signe-et-magnitude (« sign-and-magnitude »). Tous les nombres, même zéro et l'infini, ont des versions négatives.

Les nombres dans ce tableau ont été choisis afin de démontrer diverses choses :

- 0.0 : il est utile que zéro soit représenté par tous les bits à 0. Cependant, il y a aussi un zéro négatif qui a le bit de signe activé. Le zéro négatif est égal au zéro positif ;
- 1.40129846e-45 : ceci est le plus petit flottant positif et sa représentation entière est le plus petit entier positif ;
- 1.17549435e-38 : ceci est le plus petit flottant avec un chiffre 1 initial sous-entendu, le plus petit nombre avec un exposant non-zéro, le plus petit flottant normalisé. Ce nombre correspond aussi à FLT_MIN (Note de traduction : utiliser `std::numeric_limits<float>::min()` est une façon plus « C++ » de récupérer la même valeur que FLT_MIN. De même pour `std::numeric_limits<float>::max()` et FLT_MAX). Notez que FLT_MIN n'est pas le plus petit flottant. Il y a en fait à peu près 8 millions de flottants positifs plus petits que FLT_MIN ;
- 0.2 : ceci est un exemple d'un des nombreux nombres décimaux qui ne peuvent pas être représentés précisément avec un format binaire en virgule flottante. La mantisse devrait répéter C à l'infini ;
- 1.0 : notez l'exposant et la mantisse et mémorisez la représentation entière au cas où vous la verriez dans des captures hexadécimales ;
- 1.5 et 1.75 : juste quelques nombres un peu plus grands pour montrer la mantisse qui change alors que l'exposant reste identique ;
- 1.99999988 : ceci est le plus grand flottant qui a le même exposant que 1.0 et le plus grand flottant qui est plus petit que 2.0 ;
- 2.0 : notez que l'exposant est incrémenté de 1 par rapport à 1.0 et que la représentation entière et l'exposant sont tous deux plus grands que ceux de 1.99999988 ;
- 16777215 : ceci est le plus grand flottant impair. Le flottant suivant a une valeur d'exposant de 24, ce qui signifie que la mantisse est déplacée assez vers la gauche pour que les nombres impairs soient impossibles. Notez que ça signifie qu'au-dessus de 16777215, un flottant a moins de précision qu'un entier ;
- 3.40282347e+38 : FLT_MAX. Le plus grand flottant fini, avec l'exposant (non « infini ») maximum et la mantisse maximum ;
- infini positif : le papa ours des flottants.

Nous pouvons maintenant décrire comment décoder le format des flottants :

- si le champ exposant vaut 255, alors le nombre est infini (si la mantisse est nulle), ou NaN (si la mantisse n'est pas nulle) ;
- si le champ exposant vaut entre 1 et 254, alors l'exposant vaut entre -126 et 127, il y a un chiffre 1 initial sous-entendu et la valeur du flottant est :

$$(1.0 + \text{champ} - \text{mantisse}/0 \times 800000) * 2^{(\text{champ} - \text{exposant} - 127)}$$

;

- si le champ exposant est nul, alors l'exposant est -126, il n'y a pas de chiffre 1 initial implicite et la

valeur du flottant est :

$$(\text{champ} - \text{mantisse}/0 \times 800000) * 2^{-126}$$

- si le bit de signe est placé alors la valeur du flottant est l'opposé.

L'exposant sous la forme incrémentée de 127 et le chiffre 1 initial omis mènent à quelques caractéristiques très utiles des flottants, mais j'ai beaucoup de choses à dire dessus, donc souvenez-vous-en pour le prochain billet !

Retrouvez l'article de Bruce Dawson traduit par Léo Gaspard en ligne : [Lien 35](#)

Programme d'étude sur le C++ bas niveau

La série d'articles de Bruce Dawson aborde en détail les problématiques liées à la représentation des nombres à virgule flottante. Ce premier article pose les bases et explore le monde étrange et merveilleux des mathématiques à virgule flottante.

1. Contexte

Dans mon article précédent Pourquoi je suis devenu formateur ([Lien 36](#)), je déplorais le manque d'attention portée à la compréhension de la programmation bas niveau qui semble manquer aux récents diplômés en informatique (du moins en Grande-Bretagne...)

J'ai donc reçu un commentaire d'un certain Travis Woodward qui disait :

« Il y a de nombreux étudiants qui souhaitent se lancer dans du bas niveau, mais il est difficile de savoir par où commencer et quoi apprendre (le bon vieux problème de "vous ne savez pas ce que vous ignorez" ».

J'ai recherché un programme pour appréhender le bas niveau, mais n'ai trouvé que des listes de sujets qui n'aident pas à cause de l'absence de contexte ou de ressources avec lesquelles commencer. La meilleure introduction que j'ai trouvée est un cours intitulé CS107: Programming Paradigms from Stanford sur iTunes U ([Lien 37](#)), qui traite en grande partie de comment le C et le C++ sont vus du point de vue du compilateur. Donc si des programmeurs bas niveau souhaitent se regrouper afin de créer un tel cours, incluant des ressources, s'il vous plaît faites-le ! »

Comme il s'agit d'une idée louable, j'ai décidé de m'y mettre...

2. Programme d'étude bas niveau ?

Avant d'aller plus loin, je souhaiterais mettre au clair ce que j'entends par « Programme d'étude bas niveau ».

Lorsque je travaillais dans l'industrie, j'ai aidé au développement de l'architecture et au déploiement multiplateforme d'un moteur Next-Gen (qui est désormais la génération actuelle), j'ai écrit de nombreux shaders, résolu un grand nombre de bogues en étudiant le code assembleur et l'état de la mémoire de la machine, chassé les problèmes d'interblocage en milieu threadé et régulièrement travaillé sur des fichiers core dump de jeux

de tests de PS3/Xbox 360 afin de résoudre des bogues impossibles à reproduire ; mais ça ne fait pas de moi un programmeur bas niveau – c'est le genre de choses que j'attends de toute personne ayant mon expérience.

Je ne suis jamais resté assis des heures devant des captures GPad ou Pix, je ne me suis jamais vraiment soucié de corriger un fragment shader ou des calculs parallèles sur SPU ou comment tirer le meilleur de mes Altivec et je n'ai certainement jamais recodé une partie de code en assembleur en tirant parti de techniques de caches ou de mode DMA sournois pour grappiller quelques FPS – c'est ce que font les programmeurs bas niveau, du code spécifique plateforme et hardware, optimisé afin de tirer les meilleures performances d'une machine.

Ce programme d'étude n'a pas pour but de faire de vous un programmeur bas niveau.

Ce programme d'étude a pour objectif de vous faire acquérir de solides connaissances sur les fondements d'une implémentation bas niveau en C et C++ (le C sera traité comme un sous-ensemble du C++ dans cette série d'articles) – une compréhension qui selon moi devrait être la base de tout programmeur évoluant dans le jeu vidéo.

Au cours des éventuels nombreux billets de blog, j'aborderais :

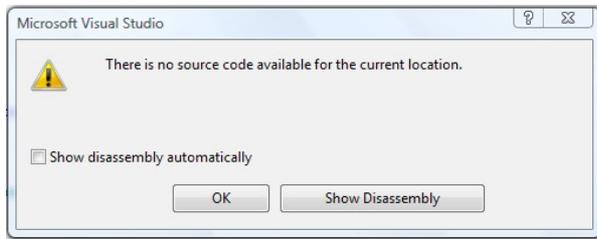
1. Les types de données ;
2. Les pointeurs, tableaux et références ;
3. Les fonctions et la pile ;
4. Le modèle objet C++ ;
5. La mémoire ;
6. La mise en cache, les différents caches.

En supposant que vous lisiez et compreniez tous les articles de cette série – et que je parvienne à communiquer les informations correctement – vous devriez atteindre un niveau qui vous permettra de connaître chaque « faiblesse » du langage, mais aussi – et c'est plus important – vous comprendrez pourquoi elles existent.

Par exemple, vous savez (peut-être pas) que les fonctions virtuelles ne peuvent pas être appelées dans un constructeur, mais avant la fin de cette série d'articles vous

comprendrez pourquoi elles ne peuvent l'être. Juste pour clarifier les choses à nouveau, je ne parle pas forcément du niveau de compréhension d'une personne qui écrit un compilateur professionnel ; mais d'un niveau qui vous donne une meilleure idée de ce qu'il se passe après l'écriture du code dans la chaîne de compilation et par conséquent vous permet de comprendre ce que le code que vous avez écrit peut impliquer.

3. Il n'y a pas de code source disponible pour l'emplacement en cours



Nooooon ! évite-moi l'hexadécimal !

Je suis sûr que la vaste majorité des programmeurs qui utilisent Visual Studio ont paniqué les premières fois qu'ils ont vu cette boîte de dialogue.

Ce fut mon cas.

J'ai appris à programmer sur un écran vert (ou orange si les écrans verts étaient déjà pris), sur un environnement Unix mainframe. Vous savez, les mêmes qu'ils ont dans les vieux films comme Alien. Les étudiants de seconde et troisième année avaient priorité pour utiliser les machines XWindows (et le peu de machines Silicon Graphics étaient réservées aux projets graphiques des étudiants de troisième année), donc j'ai appris mon métier grâce à ces Unix.

Même les machines XWindows ne possédaient pas d'EDI – j'utilisais Emacs et les makefile GNU – et le seul débogueur était GDB en ligne de commande, qui n'est pas vraiment ce qu'on appellerait « user-friendly ». J'utilisais finalement `std::cout`.

Quand j'ai été diplômé, je suis passé d'un monde aux claviers en bakélite, d'images rémanentes et lignes de commandes, au monde de Windows 95 et Visual Studio 4 (juste avant que naissent Direct X et les accélérations graphiques matérielles).

Quand j'ai vu cette boîte de dialogue pour la première fois, j'ai paniqué – et ça n'aurait pas dû arriver !

Grâce à l'enseignement focalisé sur les éléments haut niveau comme la syntaxe du langage et l'architecture du code, je n'avais aucune idée de ce qui se cachait derrière le compilateur à part ce que mes brèves incursions avec GDB m'avaient offert.

Je venais d'être diplômé d'une université très respectée où ils ne m'avaient rien enseigné du tout sur l'assembleur dans le cadre du cursus principal et j'avais supposé que c'était parce qu'ils pensaient que c'était trop pour mon esprit chétif.

Assez parlé, j'ai surmonté ma peur – mais j'ai pris cette boîte de dialogue comme un signe d'absence d'informations plus longtemps que je ne voudrais l'admettre.

Je n'ai vraiment commencé à la surmonter que quelques années plus tard, alors que je travaillais en collaboration avec quelqu'un qui avait obtenu un emploi dans l'industrie du jeu vidéo par sa connaissance de l'assembleur.

J'ai eu un crash et il a juste cliqué sur le bouton « Voir le code machine ». Il m'a alors montré, de manière naturelle,

exactement pourquoi mon code plantait – en me l'expliquant en termes de comment le C++ se traduit en code machine – et m'a indiqué comment le réparer.

Ceci m'a ouvert l'esprit sur trois points :

1. Cette personne était très décontractée face à la boîte de dialogue ;
2. Le code machine n'était clairement pas la magie noire qu'il semblait être ;
3. Ça paraissait si simple que je ne comprenais pas pourquoi ça ne nous avait pas été enseigné à l'université dans le cursus C++.

4. Déchirer le voile du code machine

Je ne réalisais pas l'importance que ça avait jusqu'à ce que je rencontre un certain Andy Yelland. Si vous connaissez Andy, vous savez exactement de quoi je parle, mais pour ceux qui ne l'ont pas rencontré, je vais expliquer.

Andy est une de ces personnes qui change votre point de vue. Il est plus ou moins l'exact opposé du stéréotype du programmeur de jeu vidéo : bien habillé, professionnel, toujours bien informé, amical, drôle et socialement intégré. Cependant, son talent le plus extraordinaire est la vitesse à laquelle il peut disséquer un fichier core dump de console. Il s'assoit calmement et étudie calmement sa pile, prenant de temps en temps des notes sur la valeur de certains registres, regardant l'adresse de la fonction dans un fichier de symbole, et au bout d'un certain temps – pouvant aller de 5 min à quelques heures selon la difficulté du problème – se retourne et vous explique exactement quel était le problème.

Non seulement il peut réaliser cet exploit sur du code qu'il n'a jamais vu auparavant – mais mieux encore, il est toujours heureux de s'asseoir et de vous expliquer toutes les actions qu'il a accomplies.

Après plusieurs dissections d'erreurs avec Andy, j'ai réalisé que ce que je prenais pour de la magie noire était en fait tout l'inverse.

Il s'agit d'avoir une connaissance experte du fonctionnement du C++ au niveau assembleur/machine et d'appliquer ces connaissances méthodiquement via l'ingénierie inversée à partir de l'état actuel (i.e. quand l'application a planté) jusqu'au point où les données incorrectes ont été introduites.

Ça demande clairement beaucoup de pratique et pour atteindre le niveau d'Andy, il faut des années (sauf pour Rain Man).

Je ne dis pas que tout le monde devrait être capable de déchiffrer le code machine d'un code écrit par quelqu'un d'autre – je n'en suis certainement pas capable.

Je pense que tout programmeur de jeu vidéo devrait être capable de regarder le code machine et être capable d'au moins comprendre ce qu'il se passe et en s'appuyant sur leur compréhension de comment le C++ est implémenté à bas niveau – peut-être avec des manuels sur le matériel à un moment donné – devrait alors être capable de comprendre l'erreur.

La première règle du cours bas niveau est : tu ne craindras pas le code machine.

Supposons que vous soyez d'accord avec moi, par où commencer ?

Je pense que le meilleur moyen est de regarder un peu de code machine, alors commençons par ça.

Faites-vous un projet de test dans votre EDI C++ préféré et écrivez ce simple code dans la fonction main.

Pour ma part j'utilise Visual Studio 2010 sur PC Windows.

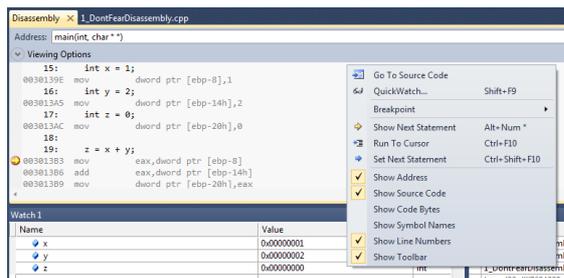
```
int x = 1;
int y = 2;
int z = 0;

z = x + y;
```

Mettez-vous en configuration « Debug » et placez un point d'arrêt sur la ligne 5 (z = x + y), puis exécutez le programme.

Quand le point d'arrêt est rencontré, faites un clic droit dans la fenêtre d'édition du code et choisissez « Code machine ».

NE PANIQUEZ PAS !



NOTE : assurez-vous d'avoir les mêmes options sélectionnées dans le menu contextuel !

Vous devriez avoir un affichage proche de l'image ci-dessus. Le texte en noir avec le numéro de ligne est notre code compilé, le code en gris sous chaque ligne montre le code assembleur généré pour chaque ligne.

5. Alors qu'est-ce que tout ça signifie ?

Le numéro en hexadécimal en début de chaque ligne d'assembleur (grise) est l'adresse mémoire de la ligne – souvenez-vous que le code n'est qu'un flux de données qui indique au CPU ce qu'il doit faire, donc logiquement il doit avoir une adresse mémoire. Ne vous inquiétez pas trop à ce sujet, je voulais juste vous montrer que les instructions sont également en mémoire.

mov et add sont des mnémoniques (mots-clés) assembleur – chacun représente une instruction, une par ligne avec ses arguments.

eax et ebp sont deux registres dans un CPU x86 (32 bits). Les registres sont des zones mémoires pour le CPU : des fragments de mémoire dans le CPU lui-même et auxquels le CPU peut accéder instantanément. Plutôt que de parler d'adresse comme pour la mémoire, les registres sont nommés en assembleur parce qu'il en existe généralement un (relativement) petit nombre.

Le registre eax est à usage général, mais est principalement utilisé pour les opérations mathématiques.

Le registre ebp est le « pointeur de base ». En assembleur x86, les variables locales seront généralement accédées via un offset à partir de ce registre. Nous verrons l'utilisation du registre ebp plus tard.

Comme mentionné dans le paragraphe précédent, ebp-8, ebp-14, et ebp-20 sont les adresses mémoires des variables locales, respectivement x, y et z, auxquelles on accède via un offset à partir de ebp.

dword ptr[...] signifie « la valeur codée sur 32 bits, stockée à l'adresse entre crochets » (ceci est vrai pour de l'assembleur Win32, ça peut différer en Win64 – je n'ai pas

vérifié (Note de traduction : ça semble également être le cas sur Visual Studio 2010 sur Windows 7 64 bits)).

6. Comment ça marche ?

Nous savons que le code assembleur généré par notre code C++ initialisera les trois variables x, y et z ; puis ajoutera x et y et stockera le résultat dans z.

Regardons chaque ligne assembleur de manière isolée (on ignorera l'adresse de la ligne).

```
mov dword ptr [ebp-8],1
```

Cette instruction initialise x en déplaçant la valeur 1 à l'adresse mémoire ebp-8.

```
mov dword ptr [ebp-14h],2
```

Cette instruction initialise y en déplaçant la valeur 2 à l'adresse mémoire ebp-14h – note : le « h » est nécessaire parce que 14 en décimal est différent de 14 en hexadécimal – ce n'était pas nécessaire pour x car 8 en décimal est également 8 en hexadécimal.

```
mov dword ptr [ebp-20h],0
```

Cette instruction initialise la valeur de z.

Maintenant nous sommes rendus à la partie intéressante, l'arithmétique pour assigner le résultat à z.

```
mov eax, dword ptr [ebp-8]
```

Cette instruction déplace la valeur à l'adresse ebp-8 (soit x) dans le registre eax...

```
add eax, dword ptr [ebp-14h]
```

... cette instruction ajoute la valeur à l'adresse ebp-14h (soit y) au registre eax...

```
mov dword ptr [ebp-20h],eax
```

... et cette instruction déplace la valeur du registre eax à l'adresse ebp-20h (soit z).

Donc, comme vous le voyez, alors que le code assembleur a l'air totalement différent, il est logiquement isomorphe au code C++ dont il est généré (i.e. son comportement peut être un peu différent mais il fournira la même sortie pour une entrée donnée).

7. Allons, pourquoi a-t-on revu ça ?

Ceux d'entre vous qui ont le cerveau connecté à leurs yeux auront remarqué que le code machine que je viens de fournir est « un peu ringard ».

Honnêtement, c'était le but en choisissant un tel exemple. Le but était de montrer comment quelque chose d'aussi simple qu'ajouter deux entiers et de stocker le résultat dans un troisième en C++ se traduisait en assembleur.

Vous pouvez utiliser cette technique pour voir comment la majorité du code C++ est traduit et généré en code machine, et le but était de montrer que c'est plutôt simple à réaliser.

Évidemment cet exemple ne montre que deux mnémoniques de l'assembleur x86 et il y en a beaucoup

plus.

Si vous rencontrez des mnémoniques que vous ne connaissez pas, il suffit généralement de les rechercher sur Google afin de connaître leur utilité et leur fonctionnement. C'est ce que j'ai toujours fait et il y a tellement d'informations sur l'assembleur x86 sur le web que vous ne devriez avoir aucun souci à déchiffrer du code.

J'ai trouvé une page très utile qui regroupe les registres x86 en détail : Lien [38](#)

Voici un lien vers une page pour télécharger un fichier PDF « antisèche » sur l'assembleur x86 : Lien [39](#)

Bien sûr la page Wikipédia : Lien [40](#)

Et un lien tiré de la page Wikipédia : Lien [41](#)

8. Résumé

Alors que peu de programmeurs auront besoin d'écrire de l'assembleur, tout programmeur de jeu vidéo aura – tôt ou tard – compris l'avantage de comprendre et déchiffrer celui-ci. C'est incroyable tout ce que vous pouvez comprendre avec une faible connaissance de l'assembleur et de sa façon de traduire le code C++.

L'exemple que nous avons vu, comme je l'ai déjà concédé, était très simple.

Le but de ce premier article n'était pas de vous donner des réponses, mais de vous montrer que le code assembleur n'est intimidant que si vous le laissez l'être ; et vous encourager à explorer ce que le compilateur fait à partir du code que vous lui fournissez.

N'abandonnez pas parce que vous ne comprenez pas ce que vous voyez ; utiliser Google ou poser des questions spécifiques aux bons endroits comme <http://stackoverflow.com> (Note de traduction : ou plutôt <http://www.developpez.net/forums/f20/autres-langages/assembleur/>).

Analyse statique de code

L'analyse statique de code permet d'améliorer la qualité d'un code et de minimiser les risques d'apparition d'erreurs. Dans cet article, John Carmack, le célèbre développeur de Doom et Quake, compare différents outils d'analyse statique de code et plus généralement ce qui fait la qualité du code.

1. Introduction

La chose la plus importante que j'ai faite en tant que programmeur ces dernières années est de poursuivre énergiquement l'analyse statique de code. Encore plus précieux que les centaines de bogues sérieux qu'elle m'a évités, c'est le changement de mentalité à propos de ma façon de voir la fiabilité logicielle et la qualité du code.

Il est important de dire d'emblée que la qualité n'est pas tout, et reconnaître ce fait n'est pas une sorte de faute morale. La « valeur » est ce que vous essayez de produire et la « qualité » n'est qu'un aspect de celle-ci, entremêlé avec le coût, les fonctionnalités et d'autres facteurs. Il y a eu de nombreux titres très appréciés et ayant obtenu un immense succès qui étaient remplis de bogues et plantaient fréquemment ; adopter le même style de développement pour un logiciel destiné à une navette spatiale et des jeux vidéo serait idiot. Pourtant, la qualité reste importante.

J'ai toujours pris soin d'écrire du bon code, l'une de mes importantes motivations internes est celle de l'artisan et j'ai toujours envie de m'améliorer. J'ai lu des tas de livres aux titres de chapitres austères comme « Politiques, normes et

9. Épilogue

Il y a d'autres points sur lesquels je voudrais attirer votre attention à partir de ce simple exemple :

1. Le principe de variable en C++ (ou de tout autre langage qui utilise des variables) n'existe pas au niveau de la machine. En code machine, les valeurs de x, y et z sont stockées à des adresses mémoires spécifiques, et le CPU y accède via leur adresse mémoire respective. Une variable est un concept haut niveau du langage, qui est plus simple à manipuler, mais est déjà un concept abstrait d'identification d'une variable dans une adresse mémoire ;
2. Notez que pour réaliser toute action « intéressante » (comme ajouter une valeur à une autre), le CPU doit avoir au moins une partie de ses données dans un registre (je suis sûr que certains CPU doivent être capables d'opérer directement sur la mémoire, mais ce n'est certainement pas la façon usuelle de les réaliser).

Finalement, je pense que cet exemple extrêmement simple illustre ce qui est pour ma part un des faits les plus importants en programmation :

Les langages haut niveau existent pour nous faciliter la vie, ils ne sont pas du tout représentatifs de la façon de fonctionner du CPU – en fait, même l'assembleur est une commodité à côté du code binaire que les mnémoniques (mov, add, etc.) représentent.

Je vous conseille de ne pas trop vous inquiéter du code binaire, et souciez-vous encore moins des électrons et de la silicium.

Retrouvez l'article d'Alex Darby traduit par Bousk en ligne : Lien [42](#)

plans qualité », et mon travail avec Armadillo Aerospace m'a mis en contact avec le monde très différent du développement de logiciel critique.

Il y a dix ans, lors du développement de Quake 3, j'ai acheté une licence pour PC-lint et essayé de l'utiliser – l'idée de pointer automatiquement les erreurs dans mon code avait l'air excellente. Cependant, l'utiliser en ligne de commande et passer au crible les monceaux de commentaires qu'il produit n'a pas réussi à me convaincre, et je l'ai abandonné assez rapidement.

Le nombre de programmeurs et la taille du code ont tous les deux augmenté d'un ordre de grandeur depuis et le langage d'implémentation est passé du C au C++, ce qui constitue un terrain d'autant plus fertile pour les erreurs logicielles. Il y a quelques années, après avoir lu un certain nombre d'articles de recherche sur l'analyse de code statique moderne, j'ai décidé de voir comment les choses avaient changé dans la décennie depuis que j'avais essayé PC-lint.

À ce stade, nous compilons au niveau de warning 4 en désactivant seulement quelques warnings très spécifiques,

et le mode warning-as-error forçait les programmeurs à s'y conformer. Malgré la présence de quelques tronçons de code poussiéreux ayant accumulé des années de cochonneries, la plupart du code était assez moderne. Nous pensions avoir une assez bonne base de code.

2. Coverity

Au départ, j'ai pris contact avec Coverity ([Lien 43](#)) et signé pour une période d'essai. C'est un logiciel sérieux, avec des coûts de licence basés sur le nombre total de lignes de code et nous nous sommes retrouvés avec une estimation à cinq chiffres. Quand ils ont présenté leur analyse, ils ont dit que notre base de code était l'une des plus propres de cette taille qu'ils avaient vu (peut-être qu'ils disent cela à tous les clients pour qu'ils se sentent de bonne humeur), mais ils ont présenté un ensemble d'une centaine de problèmes qui avaient été identifiés. C'était très différent de l'ancien PC-lint. Le ratio signal/bruit était très élevé - la plupart des problèmes soulevés étaient du code manifestement erroné qui pouvait avoir des conséquences graves.

Cela nous a ouvert les yeux, mais le coût était assez élevé pour qu'il nous donne à réfléchir. Il nous restait plus qu'à espérer que le nombre de nouvelles erreurs que nous allions introduire ne soit pas trop important avant de livrer le jeu.

3. Microsoft /analyze

Je me serais probablement convaincu de finalement acheter Coverity, mais alors que j'étais encore en train d'en débattre, Microsoft préempta le débat en intégrant leur fonctionnalité /analyze dans le SDK 360. /analyze ([Lien 44](#)) était auparavant disponible pour la version haut de gamme, ridiculement chère de Visual Studio, mais il était maintenant disponible pour tous les développeurs 360 sans supplément. Mon interprétation est que Microsoft estime que la qualité des jeux sur la 360 a plus d'impact que la qualité des applications Windows.

Techniquement, l'outil de Microsoft effectue seulement une analyse locale, il devrait donc être inférieur à l'analyse globale de Coverity, mais il a suffi de l'activer pour qu'il déverse une montagne d'erreurs, beaucoup plus que ce que Coverity avait signalé. Certes, il y avait beaucoup de faux positifs, mais il y avait aussi beaucoup de choses vraiment, vraiment effrayantes.

J'ai progressé lentement dans le code, en corrigeant en premier lieu mon code personnel, puis le reste du code système, puis le code du jeu. J'y travaillais pendant des bouts de temps libre, de sorte que le processus s'est étendu sur deux ou trois mois. L'un des avantages secondaires de cette longue période est d'avoir montré de manière concluante qu'il signalait des choses très importantes - durant ce temps il y avait eu une chasse au bogue épique, mobilisant de nombreux programmeurs pendant de nombreux jours et qui a fini par être attribuée à quelque chose que /analyze avait signalé, mais que je n'avais pas encore corrigé. Il y a eu plusieurs autres cas, moins dramatiques, où le débogage a conduit directement à quelque chose déjà signalé par /analyze. C'était des problèmes réels.

Finalement, j'ai eu tout le code utilisé pour générer l'exécutable 360 compilant sans warnings avec /analyze, je l'ai alors établi comme comportement par défaut pour les builds 360. Chaque programmeur travaillant sur la 360 a

ensuite vu son code analysé à chaque fois qu'il lançait un build, de sorte qu'il remarquait par lui-même les erreurs qu'il venait de commettre, plutôt que ce soit moi qui les corrige silencieusement plus tard. Cela ralentissait la compilation quelque peu, mais /analyze est de loin l'outil d'analyse le plus rapide avec lequel j'ai travaillé, et il en vaut tellement la peine.

Nous avons eu une période où, sur l'un des projets, l'option d'analyse statique fut accidentellement désactivée pendant quelques mois, et quand je l'ai remarquée et réactivée, il y avait eu des tas de nouvelles erreurs introduites dans l'intervalle. De même, les programmeurs travaillant seulement sur PC ou PS3 archivaient du code défectueux et ils ne le réalisaient pas jusqu'à ce qu'ils obtiennent un rapport « build 360 échoué » par mail. C'était des démonstrations que le processus normal de développement produit constamment ces classes d'erreurs et /analyze nous avait effectivement protégé de nombre d'entre elles.

Bruce Dawson a bloqué sur le travail avec /analyze un certain nombre de fois : code reliability ([Lien 45](#)).

4. PVS-Studio

Parce que nous utilisons /analyze uniquement sur le code pour 360, nous avons encore beaucoup de codes non couverts par l'analyse - les codes spécifiques pour PC et PS3, ainsi que tous les utilitaires tournant uniquement sur PC.

L'outil suivant sur lequel j'ai porté mon attention était PVS-Studio : [Lien 46](#). Il s'intègre bien avec Visual Studio, et possède un mode de démonstration pratique (essayez-le !). Par rapport à /analyze, PVS-Studio est douloureusement lent, mais il a détecté un certain nombre d'autres erreurs importantes, même dans du code qui était déjà complètement propre avec /analyze. En plus de détecter des erreurs logiques, PVS-Studio détecte aussi un certain nombre de modèles communs d'erreurs de programmation, même si ça reste du code tout à fait raisonnable. Cela garantit de produire des faux positifs, mais que je sois damné si nous n'avions pas des instances de ces patterns d'erreurs communs nécessitant des corrections.

Il y a un certain nombre de bons articles sur le site de PVS-Studio ([Lien 47](#)), la plupart avec des exemples de codes tirés de projets open source montrant exactement le genre de choses que l'on trouve. J'ai envisagé d'ajouter des exemples représentatifs d'avertissements d'analyse de code dans cet article, mais il y a déjà des exemples mieux documentés présentés là-bas. Allez les voir, et ne vous moquez pas en pensant : « jamais je n'écrirais ça ! ».

5. PC-lint

Enfin, je suis retourné à PC-lint ([Lien 48](#)), couplé avec Visual Lint ([Lien 49](#)) pour intégration dans l'IDE. Dans la grande tradition Unix, il peut être configuré pour faire à peu près n'importe quoi, mais ce n'est pas très amical, et on est en général loin du « clé en main ». J'ai acheté un pack de cinq licences, mais cela a été si problématique qu'il me semble que tous les autres développeurs ayant essayé ont abandonné. La flexibilité présente des avantages - j'ai pu le configurer pour analyser l'ensemble de notre code spécifique à la plate-forme PS3, mais ce fut un travail pénible.

Une fois de plus, même dans du code qui avait été nettoyé à la fois par /analyze et PVS-Studio, de nouvelles erreurs

significatives ont été trouvées. J'ai fait un réel effort pour nettoyer notre base de code avec Lint, mais je n'ai pas réussi. Je l'ai fait pour le code système, mais me suis essoufflé face à tous les rapports dans le code du jeu. J'ai trié en m'occupant en priorité des classes de rapports qui m'inquiétaient le plus, et en ignorant l'essentiel des rapports qui étaient plus d'ordre stylistique ou concernant des soucis potentiels.

Essayer de moderniser une base de code importante pour être propre aux niveaux maximums offerts par PC-lint est probablement futile. J'ai fait un peu de programmation « feu vert » où j'ai servilement traité un par un chaque commentaire Lint, mais c'est un ajustement que la plupart des programmeurs expérimentés C/C++ ne vont pas vouloir faire. J'ai encore besoin de passer un peu de temps à essayer de déterminer le bon ensemble de warnings à activer pour permettre de tirer le meilleur profit de PC-lint.

6. Discussion

J'ai appris beaucoup de choses grâce à ce processus. Je crains que certaines de ces choses ne soient pas facilement transmissibles. Et que sans parcourir personnellement des centaines de rapports, en très peu de temps et en ressentant encore et encore ce nœud dans l'estomac, « tout est OK » ou « ce n'est pas si mal » seront les réponses par défaut.

La première étape est d'admettre sans réserve que le code que vous écrivez est truffé d'erreurs. C'est une pilule amère à avaler pour beaucoup de gens, mais sans elle, la plupart des propositions de changement seront vues avec irritation voire franche hostilité. Vous devez chercher à avoir des critiques de votre code.

L'automatisation est nécessaire. Il est fréquent d'avoir une sorte de satisfaction suffisante en entendant des récits de faillites retentissantes des systèmes automatiques, mais pour chaque échec de l'automatisation, les échecs de l'homme sont légion. Les exhortations à « écrire du meilleur code », les projets de revue de code, de programmation en binôme, et ainsi de suite, ne sont tout simplement pas suffisants, en particulier dans un environnement avec des dizaines de programmeurs travaillant sur des délais très courts. La valeur qu'il y a à capturer même le plus petit sous-ensemble d'erreurs accessibles à l'analyse statique à tous les coups est immense.

J'ai remarqué qu'à chaque fois que PVS-Studio était mis à jour, il trouvait quelque chose dans notre base de code avec les nouvelles règles. Cela semble vouloir dire que si vous avez une base de code suffisamment grande, toute classe d'erreur qui est syntaxiquement légale existe probablement là-dedans. Dans un grand projet, la qualité du code est tout aussi statistique que les propriétés physiques des matériaux – des défauts existent partout, on ne peut qu'espérer minimiser l'impact qu'ils ont sur les utilisateurs.

Les outils d'analyse travaillent avec une main attachée dans le dos, en étant obligés de déduire des informations à partir de langages qui ne fournissent pas nécessairement ce qu'ils veulent, et en faisant généralement des hypothèses très conservatrices. Vous devez collaborer autant que possible – favorisez l'indexation plutôt que l'arithmétique de pointeurs, essayez de garder votre graphe d'appel dans un seul fichier source, utiliser les annotations explicites, etc. Tout ce qui n'est pas clair pour un outil d'analyse statique n'est probablement pas clair non plus pour vos

collègues programmeurs. Le dédain classique du hacker pour la rigueur et les contraintes est une vision à court terme – les besoins des gros projets ayant une longue durée de vie et de multiples développeurs sont simplement différents du travail rapide que vous faites pour vous-même.

Les pointeurs NULL sont la plus grande source d'erreur en C/C++, du moins dans notre code. L'utilisation duale d'une valeur unique à la fois comme un flag et comme une adresse provoque un nombre incroyable de problèmes fatals. En C++ les références doivent être préférées par rapport aux pointeurs chaque fois que c'est possible. Bien qu'une référence ne soit « en réalité » qu'un pointeur, elle possède le contrat implicite de ne pas être NULL. Faites le check du NULL au moment de transformer un pointeur en référence et vous pourrez par la suite ignorer le problème. Il y a de nombreux patterns profondément enracinés dans la programmation de jeu vidéo qui sont tout simplement dangereux, mais je ne suis pas sûr de savoir comment migrer en douceur loin de tous ces check de NULL.

Les erreurs de formatage de printf étaient la deuxième plus grande source d'erreur dans notre base de code, accentuées par le fait que le passage d'une `idStr` au lieu d'un `idStr::c_str()` entraîne presque toujours un crash, cependant annoter toutes nos fonctions variadiques avec les annotations de `/analyze` pour que les types soient correctement contrôlés a tué ce problème pour de bon. Il y avait des dizaines d'erreurs de ce style, cachées dans des messages de warnings riches en enseignement qui se transformaient en crash quand certaines conditions inhabituelles activaient le chemin d'exécution, ce qui est aussi un commentaire sur la façon dont la couverture du code de nos tests en général faisait défaut.

Un grand nombre des erreurs graves rapportées sont dues à des modifications dans du code longtemps après qu'il a été écrit. Un motif d'erreur incroyablement courant est d'avoir un bout de code en parfait état qui vérifie un pointeur NULL avant de faire une opération, mais une modification du code plus tard change de sorte que le pointeur est utilisé à nouveau sans vérifier. Examiné séparément du contexte, ceci peut être un argument en faveur de la complexité du code, mais quand vous revenez sur le déroulement de ce qui c'est passé, il est clair que cela provient plus d'un manque de communication sur les préconditions d'utilisation avec le développeur qui modifie le code.

Par définition, vous ne pouvez pas vous concentrer sur tout, donc concentrez-vous sur le code qui va être livré aux clients, plutôt que sur le code qui sera utilisé en interne. Migrer de force le code destiné à être livré sur des projets de développement isolés. Un article récent a remarqué que toutes les différentes métriques de qualité du code étaient corrélées au moins aussi fortement avec le taux d'erreurs que la taille du code, ce qui donne à la taille du code seule essentiellement la même capacité de prédiction du taux d'erreur. Réduisez la taille de votre code important.

Si vous n'êtes pas profondément effrayés par tous les problèmes additionnels soulevés par la programmation parallèle, vous n'avez pas réfléchi suffisamment à ce sujet. Il est impossible de faire du vrai contrôle de test dans le développement logiciel, mais je crois que le succès que nous avons eu avec l'analyse de code a été suffisamment clair pour que je puisse le dire sans détour : il est irresponsable de ne pas l'utiliser. Il y a des données objectives dans les rapports automatiques des crashes de console montrant que Rage, en dépit d'être à la pointe de la

technologie à bien des égards, est remarquablement plus robuste que la plupart des titres contemporains. Le lancement PC de Rage a malheureusement été tragiquement entaché par des problèmes de pilote - je parierais qu'AMD n'utilise pas l'analyse statique de code sur leurs pilotes graphiques.

Ce qu'il faut retenir : si votre version de Visual Studio possède /analyze, activez-le et essayez-le. Si je devais choisir un seul outil, je choisirais l'option Microsoft. Pour tous ceux qui travaillent avec Visual Studio, essayez au

moins la démonstration de PVS-Studio. Si vous développez des logiciels commerciaux, l'achat d'outils d'analyse statique est de l'argent bien dépensé.

Un dernier commentaire venant de Twitter : Dave Revell @dave_revell : « Plus je passe du code à l'analyse statique, plus je m'émerveille que les ordinateurs démarrent tout court. »

Retrouvez l'article de John Carmack traduit par Arzar en ligne : [Lien 50](#)

Les modules de Qt 5

L'un des principaux changements que l'on trouvera dans Qt 5 est la réorganisation des modules. Les modules sont regroupés en deux groupes : les *Essentials*, installés automatiquement, et les *Add-ons*, installés à la demande. Puisque Qt 5 n'est pas encore en version finale, les informations données dans cet article sont susceptibles d'être modifiées.

Les modules Essentiels

Le module Qt Core

Ce module fournit les fonctionnalités de base de Qt, excepté ce qui concerne l'interface graphique. Tous les autres modules sont liés à ce module. Voici la liste de quelques ajouts dans Qt 5 :

- *QStandardPaths* : permet de récupérer les répertoires par défaut en fonction de la plateforme. C'est une évolution de *QDesktopServices* avec plus de fonctionnalités, sur le modèle de *KStandardDirs* de KDE. Cela permet par exemple de faire une recherche de toutes les occurrences d'un fichier dans les différents répertoires ;
- support de JSON : permet de créer ou de lire un fichier JSON à partir d'une représentation binaire en mémoire ;
- extension prise en charge MIME : permet de déterminer le type mime d'un fichier ou de données en mémoire, en fonction de l'extension et/ou du contenu. Ce module utilise une base de données des types MIME par *QMimeDatabase* fourni par freedesktop.org shared-mime-info project. Cette base de données est incluse par défaut dans le système sous Linux et fourni par Qt sur Windows et Mac OS X ;
- vérification des connexions signaux/slots à la compilation : vérifie l'existence du signal et du receveur et que les arguments sont compatibles. Cette fonctionnalité utilise les templates et est compatible avec C++11. Il est possible de connecter un signal à des fonctions lambda, des fonctions membres ou des fonctions statiques, sans avoir besoin de déclarer comme slots. Voir l'article détaillé sur le sujet : Les signaux et slots dans Qt 5 ([Lien 51](#));
- *QRegularExpression* : nouveau moteur d'expressions régulières compatible Perl, plus puissante et rapide que *QRegExp*, avec plus de fonctionnalités (lazy and possessive quantifiers, lookbehinds, named capturing groups and iteration of matches) ;
- amélioration des performances, en particulier pour les structures de données ;
- amélioration du support C++11 quand c'est possible (mais compatibilité avec C++98) ;

- support des boutons supplémentaires sur les souris (souris pour joueurs), jusque 27 boutons pour XCB, XLIB ou DirectFB, jusque 16 pour Wayland, Evdev ou OS-X, jusque 8 pour BlackBerry/QNX et 5 sur Windows (limitation due au système).

Le module Qt Gui

Ce module fournit les fonctionnalités de base pour créer une interface utilisateur. Les anciennes classes *QWidget* et dérivées sont séparées dans un module indépendant. Ce module contient de nouvelles classes comme *QWindow*, *QScreen*, *QSurfaceFormat* permettant le support de fonctionnalités de base et est surtout destiné à être utilisé par les autres modules (*QWidget*, *QQuickView*, *Qt 3D View*, etc.).

En particulier, ce module fournit les classes *QOpenGLxxx* (*QOpenGLFramebufferObject*, *QOpenGLShaderProgram*, *QOpenGLFunctions*, *QOpenGLContext*, etc.) permettant de fournir l'accélération matérielle pour tous les modules graphiques (widgets traditionnels, Qt Quick). La classe *QOpenGLContext* est plus générique que *QGLContext* et est découplée de *QWindow*, pour permettre d'utiliser un contexte commun pour plusieurs affichages. *QOpenGLPaintDevice* permet d'utiliser directement *QPainter* sur un contexte OpenGL sans devoir dériver de *QWindow* ou *QOpenGLFramebufferObject*.

Voir l'article OpenGL dans Qt 5 pour plus de détails : [Lien 52](#).

Le module Qt QML

Ce module permet d'utiliser le langage de script déclaratif QML grâce au *QML engine*. Il présente des améliorations des performances et des ajouts de fonctionnalités par rapport celui inclus dans Qt 4.

Le module Qt Js backend (JavaScript)

Ce module fournit un interpréteur JavaScript, permettant de scripter les applications écrites en C++ et en QML. Il utilise un nouveau moteur JS v8 ([Lien 53](#)) plus rapide. Il inclut de nouvelles classes (*QJSEngine*, *QJSValue*), le support de nouveaux types (*QColor* avec les propriétés r, g, b et a, *QVector4D* constructible avec *Qt.vector4d()*). Il est possible d'ajouter des fonctionnalités dans un namespace avec la fonction *qmlRegisterModuleApi* et d'importer du QML et du JS directement dans un fichier JS.

Le module Qt Quick

Ce module permet de créer des interfaces dynamiques riches, en utilisant les modules QML et le JS. Cette nouvelle version de Qt Quick correspond au module « qtquick2 » alors que l'ancienne version correspond au module « qtquick1 ». L'interface graphique de Qt Quick 2 se base maintenant sur scenegraph et permet l'accélération matérielle en utilisant les classes *QOpenGLxxx* de Qt Gui. On trouve de nouvelles classes (*QQuickView*, *QQuickCanvas*, *QQuickItem* et *QQuickPaintedItem* qui

remplacent les classes équivalentes de *QDeclarative*), de nouveaux items (*Canvas* permet le support de l'API *Context2D* de HTML 5, le rendu est réalisé dans *Canvas.Image* et *Canvas.FramebufferObject*, avec support multithread en arrière-plan). Le moteur de particules 2D *Qt Quick.Particles 2.0* et la collection d'effets de shaders qui étaient avant des projets séparés dans *Qt Labs* sont maintenant inclus dans *Qt*.

Le module Qt 3D

Le module *Qt 3D* est également un ancien projet provenant de *Qt Labs* et est inclus dans *Qt 5*. Il a permis dans *Qt 4* l'ajout de nombreuses fonctionnalités de calculs 3D comme les classes *QMatrix4x4*, *QGLShaderProgram* et *QVector3D*. Il utilise en interne le module *Qt QML* et le support *OpenGL* de *Qt Gui*. Ce module contient deux bibliothèques : *Qt 3D* (pour utiliser directement la 3D en C++) et *Qt 3D Quick* (pour l'utilisation dans *Qt Quick*).

Plusieurs fonctionnalités sont ajoutées :

- gestion de scènes 3D, avec rendu en *OpenGL* ;
- lecture de fichiers 3D (par exemple *.obj* et *.3ds*) ;
- gestion des lumières, des meshes, des textures, des matériaux, des animations, des caméras, des vues ;
- ajout de shader directement ou par fichier dans les propriétés *QML* ;

Le module Qt Location

Ce module est un ajout dans *Qt 5*, mais il existait déjà depuis des années comme sous-ensemble de *Qt Mobility*. Il fournit les services nécessaires pour la localisation : *GPS*, cartographie, etc.

Il inclut une fonctionnalité permettant d'afficher des cartes avec *MapQuickItem*. L'affichage se base sur une approche modèle/vue et bénéficie de l'accélération *OpenGL* dans *scenegraph*. Les gestuelles pour les zooms et les panoramas dynamiques, le routage et le géocodage, l'ajout de repères sur les cartes sont pris en charge.

Le module Qt Network

Ce module fournit une interface portable pour utiliser les réseaux. Parmi les évolutions :

- amélioration du support *IPv6* et des réseaux utilisant les types d'adresse *IP*, de manière transparente par défaut. En réception, *QTcpServer* et *QUdpSocket* lancés avec *QHostAddress::Any* permet de recevoir dans les deux modes ; avec *QHostAddress::AnyIPv4* et *QHostAddress::AnyIPv6* permet de travailler sur un mode uniquement. En émission, *QNetworkAccessManager* tente d'utiliser les deux modes et garde le premier qui réussit ;
- *QTcpSocket* peut maintenant être attaché à un socket existant avant de lancer une connexion, pour limiter les connexions dans un environnement multihôte ;
- *QDnsLookup* permet de rechercher des enregistrements *DNS*. Il ne remplace pas *QHostInfo*, qui permet de résoudre les noms en adresse *IP*, mais permet principalement d'utiliser les autres types d'enregistrements *DNS* : *SRV*, *TXT* et *MX* ;
- les classes *QFtp* et *QHttp* ne sont pas conservées dans ce module, mais restent disponibles dans un

module indépendant pour la compatibilité. Elles sont remplacées par *QNetworkAccessManager* ;

- extensions et vérifications des certificats *SSL* : prise en charge des extensions des certificats. La vérification des certificats ne se fait plus uniquement lors de la connexion à un serveur ;
- support des clés privées masquées : permet de lire une clé privée à partir d'un périphérique, par exemple un dongle *PKCS#11*.

Les autres modules

- *Qt Multimedia* : fournit les fonctionnalités de base pour lire l'audio, la vidéo, la radio et gérer les caméras ;
- *Qt SQL* : fournit une prise en charge portable des bases de données *SQL* ;
- *Qt Test* : fournit les outils nécessaires pour implémenter des tests unitaires ;
- *Qt WebKit* : basé sur *WebKit 2*, mais sans changement de l'API C++. Ce module continue l'amélioration de prise en charge *HTML 5* et des performances.

Les modules Add-ons

Le module Qt Widget

Ce module fournit l'ensemble des classes *QWidget* et dérivées pour la compatibilité avec *Qt 4*. Il utilise la nouvelle architecture *Qt Platform Abstraction (QPA)* : Lien [54](#).

Le module Qt Quick 1

Ce module permet d'utiliser la version de *Qt Quick* disponible dans *Qt 4*, pour compatibilité. Pour utiliser ce module, il suffit d'ajouter dans le *.pro* :

```
QT += quick1
```

Et d'inclure les fichiers d'en-têtes :

```
#include QtQuick1/QDeclarativeView
#include QtQuick1/QDeclarativeItem
```

Les autres modules

- *Qt Script* : permet de rendre les applications scriptables, compatibilité avec *Qt 4*. Il utilise les classes *QJSxxx* du module *Qt Js* ;
- *Qt Bluetooth* : prise en charge du *Bluetooth* ;
- *Qt D-Bus* : interprocessus communication avec *D-Bus* ;
- *Qt Graphical Effects* : collection d'effets graphiques ;
- *Qt Image Formats* : prise en charge de formats d'images supplémentaires (*TIFF*, *MNG*, *TGA*, *WBMP*) ;
- *Qt OpenGL* : module 3D *OpenGL* compatible avec *Qt 4* ;
- *Qt Print Support* : support pour l'impression ;
- *Qt Publish and Subscribe* ;
- *Qt Script Tools* : outils supplémentaires pour scripter ;
- *Qt Sensor* : gestion des capteurs (accéléromètre, détecteur de lumière ambiante, compas, etc.) ;

- Qt Service Framework : permet de fournir des services en ligne ;
- Qt SVG : prise en charge du format d'image SVG (image vectorielle) ;
- Qt System Info : informations sur le système (profil utilisateur, batterie, stockage, etc.) ;
- Qt Tools : outils divers (Qt Designer, Qt Help, etc.) ;
- Qt Pim : contacts, organisateur, vCard, etc. ;
- Qt WebKit Widgets : version de wekbit 1, pour compatibilité avec Qt 4 ;
- Qt XML : fichier XML avec SAX et DOM. Ce module est déprécié, il faut maintenant utiliser QXmlStreamReader/Writer ;
- Qt XML Patterns : support pour XPath, XQuery,

XSLT et XML Schema validation.

Les modules accessoires

Le support de ces modules n'est pas encore déterminé.

- Active Qt : support des ActiveX et Com (Windows) ;
- Qt Feedback ;
- Qt JSON DB ;
- Phonon : support du framework phonon pour la vidéo et audio ;
- Qt QA : auto test, pour gestion automatique des tests ;
- Qt QLALR : interpréteur LALR.

Retrouvez ce billet blog de Guillaume Belz en ligne : [Lien 55](#).

Les dernières news

Sortie de Qt 5 alpha

La première version majeure du Qt Project autonome se concentre sur les performances et les capacités graphiques

La version 5 de Qt vient de sortir en version alpha. Cette version est la première version majeure depuis que Qt est devenu autonome avec la création du Qt Project. Beaucoup de personnes ont contribué à cette nouvelle version, pas uniquement des développeurs de chez Nokia. Les différents modules ont été regroupés en deux catégories, les essentiels, installés par défaut, et les add-ons, installés à la demande. L'objectif de cette version alpha est de récupérer les retours des utilisateurs, principalement sur les modules essentiels.

Lars Knoll, le responsable en chef du projet Qt, a publié en mai dernier deux discussions sur les QtLabs pour présenter les approches choisies pour Qt 5 (voir les discussions **Thoughts about Qt 5** ([Lien 56](#)) et **Responses to Qt 5** ([Lien 57](#))). La pensée directrice est résumée dans les phrases suivantes :

« Qt 5 doit être le fondement d'une nouvelle façon de développer des applications. Tout en offrant la puissance de Qt natif en C++, l'accent sera mis sur un modèle où le C++ sera principalement utilisé pour implémenter des fonctionnalités modulaires d'arrière-plan pour Qt Quick », a déclaré Lars Knoll.

Neuf mois de travail, plusieurs centaines d'intervenants et plusieurs milliers de modifications du code ont été nécessaires pour aboutir à cette version alpha. Pour cette première version majeure, l'accent a été mis sur la partie embarquée, proche de la vision que Lars Knoll a décrite, mais il faudra attendre les versions 5.1 ou 5.2 pour que cette vision soit entièrement appliquée pour la version desktop.

Cette version alpha est l'aboutissement d'un travail important sur quatre points : QPA, la pile graphique, la modularité et le nettoyage de l'architecture en déplaçant les QWidgets dans les modules add-ons.

Le Qt Platform Abstraction Layer (QPA) ([Lien 54](#))

Pour améliorer la portabilité de Qt, il a été nécessaire de restructurer l'architecture pour isoler toutes les fonctionnalités de bas niveau qui sont spécifiques à une plateforme. Ce travail a permis d'aboutir au QPA, facilitant le portage de Qt sur toutes nouvelles plateformes. Cette abstraction a été introduite dans Qt 4.8 en remplacement de QWS pour les versions embarquées de Qt, mais elle est maintenant disponible pour toutes les éditions dans Qt 5. La meilleure preuve de l'efficacité de cette abstraction est que plusieurs portages sont en cours de développement : pour QNX, iOS et Android, par exemple.

La réorganisation de la pile graphique

Un autre objectif majeur pour Qt 5 est l'amélioration des performances graphiques, en particulier pour les versions embarquées. Pour ce faire, il a fallu réorganiser la pile graphique, pour bénéficier au maximum de l'accélération matérielle. Pour cela, l'accent a été mis sur l'utilisation d'OpenGL.

Par exemple, QtQuick 2 a subi une réorganisation importante se basant sur le graphe de scène et utilisant OpenGL (GL ES 2 minimum) en arrière-plan. QtGui contient maintenant des classes QOpenGL à la place des classes QGL (maintenues dans le module QtOpenGL pour la compatibilité).

On note l'apparition de nouvelles classes :

- QGuiApplication, plus légère que QApplication (hérite de QCoreApplication et dérivée par QApplication) ;
- QWindow, pour manipuler les fenêtres de premier plan. QWidget et dérivées continuent de fonctionner, comme dans Qt 4, avec QPainter, bien que cet outil soit moins utilisé pour les autres piles graphiques (il est maintenant limité à la rasterisation logicielle sur écran, les images et les pixels, avec un backend OpenGL et un autre pour la génération de PDF et l'impression).

L'architecture modulaire

Objectif : flexibilité, possibilité de choisir ses modules pour les utilisateurs, meilleure intégration de QtMobility, faciliter les contributions en les incluant comme modules tiers. Il s'agit principalement de ménage interne, peu visible par les utilisateurs (toujours en cours).

Déplacer QWidget dans un module indépendant

Déplacer ces classes dans le module "widgets" permet de garantir la continuité des QWidget et dérivés, mais également l'évolution vers d'autres approches (QML et QtQuick). Cela nettoie l'architecture sur le long terme.

Installation et compilation

Il y a plusieurs moyens d'installer Qt 5. Le plus simple est d'utiliser les binaires non officiels, régulièrement mis à jour :

- à partir des dépôts ppa (Linux) : Lien [58](#) ;
- à partir de Git : Building Qt 5 from Git ([Lien 59](#)) ;
- à partir des sources Qt 5.0 Alpha release en différents formats (7z, tar.bz2, tar.gz, tar.xz, zip) : Lien [60](#) ;
- pour compiler : Qt 5 Alpha building instructions ([Lien 61](#)).

Passer de Qt 4 à Qt 5

Les changements importants pour conserver la compatibilité du code écrit pour Qt 4 avec Qt 5 sont d'intégrer le module widgets si on utilise des QWidget ou dérivés et de renommer le module QtQuick en quick1. Voici un exemple de code dans le fichier .pro pour garantir la compatibilité :

```
greaterThan(QT_MAJOR_VERSION, 4)
{
    QT += widgets
    QT += quick1
} else {
    QT += declarative
}
```

Le script Perl `qtbases/bin/fixqt4headers.pl` met à jour les inclusions des fichiers d'en-tête.

Pour la création de plugins, les macros `Q_EXPORT_PLUGIN` et `Q_EXPORT_PLUGIN2` sont dépréciées et doivent être remplacées par la macro `Q_PLUGIN_METADATA`, qui permet de lire les informations sans devoir charger le plugin avec la fonction `dlopen()`.

Commentez la news de Guillaume Belz en ligne : [Lien 62](#)

Premier jour de Qt chez Digia, après son rachat de Nokia : le framework restera disponible sous GPL et LGPL

Ce 18 septembre 2012 était le premier jour avec Digia propriétaire de Qt. Avec une équipe renforcée (notamment par Lars Knoll), Digia annonce que le « vrai potentiel de Qt sera libéré, ainsi que son écosystème ». Dès l'annonce du rachat en août dernier, une série de questions se posaient sur l'avenir de l'écosystème libre : Qt sera-t-il toujours disponible sous la LGPL ? Quid de Necessitas, pour le support d'Android, puisqu'un objectif était le support tant d'iOS et d'Android ?

L'objectif de Digia est toujours de « garder Qt disponible tant en commercial qu'en open source, tout en continuant à

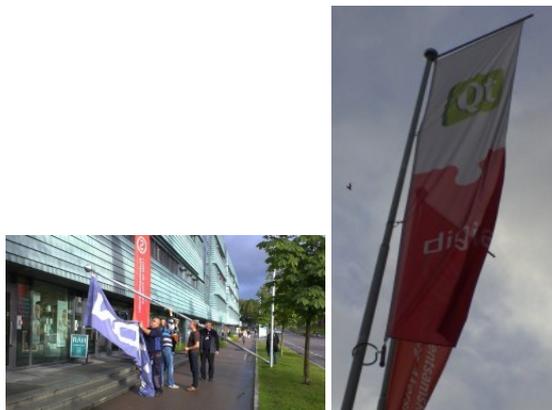
maintenir activement les deux », d'« avoir une communauté Qt forte et unie », sachant que « faire avancer la technologie est extrêmement important ».

La transaction comprenait également le financement et la gestion de l'infrastructure du Qt Project ; ce transfert est actuellement presque achevé, ne restent que quelques items (comme le système d'intégration continue).

Transfert du copyright

En pratique, qu'est-ce que cela change que la technologie, le copyright et les marques Qt soient transférés à Digia ? Il faut adapter tous les en-têtes des fichiers sources ; aussi, Digia devient le seul interlocuteur, quelle que soit la licence choisie (commerciale ou open source). À ce sujet, **Qt reste disponible tant sous la GPL que sous la LGPL**, en ce qui concerne l'open source. Également, un processus de définition d'un programme de partenaires et d'autres est en cours d'élaboration pour réguler l'utilisation des logos Qt et des autres marques déposées.

Cela implique notamment le changement du drapeau à proximité des bureaux :



Répercussions sur le Qt Project

Selon le principe de gouvernance ouverte (ou de démocratie, à d'autres échelles), le Qt Project est hébergé par une fondation à but non lucratif : Digia fournit les moyens financiers nécessaires pour son bon fonctionnement. Le fonctionnement des services n'en sera pas changé, à une exception près.

Le système d'intégration continue est toujours hébergé chez Nokia (il s'agit d'un garde-fou pour la qualité de Qt : une combinaison d'outils de grande capacité de compilation et de procédures pour s'assurer que toute modification sur Qt fonctionne correctement) ; en même temps que le transfert chez Digia, un changement du logiciel utilisé est prévu : Jenkins sera utilisé, pour faciliter l'ajout de nouvelles plateformes de validation par des agents extérieurs. Le changement devrait être achevé dans les mois ; pendant ce temps, le système actuel continuera de fonctionner.

Plateformes mobiles

Un des objectifs clairement annoncés depuis le rachat pour Qt 5 est le support de plus de plateformes mobiles – notamment, iOS et Android –, de telle sorte que tous aient

accès à ce support (**tant en commercial... qu'en open source**).

Ce support a déjà été initié par la communauté, par le projet Necessitas (KDE) pour Android par exemple. Dès aujourd'hui, cette solution peut être utilisée par tous ; cependant, des discussions ont eu lieu entre les développeurs de Necessitas et Digia, les deux parties s'accordant sur le fait qu'il ne pourrait être qu'une bonne chose de continuer le développement de Necessitas sous l'ombrelle du Qt Project, pour suivre la tendance de KDE envers Qt.

En ce qui concerne iOS, les Qt Labs présentaient une preuve que le support est possible (Lien [63](#)) ; il existe également une solution (entièrement commerciale) de Mediator Software, bien que pas aussi complète que pour Android. Des discussions sont en cours pour voir s'il est possible d'utiliser cette solution pour le Qt Project. De même, il faudra étudier les restrictions imposées par iOS qui retardent le support de Qt Quick.

Le support dans le Qt Project pour ces plateformes est prévu pour Qt 5.1 (deuxième trimestre de 2013), bien qu'il soit déjà possible de développer des applications pour ces plateformes avec Qt 4.8 et 5.0.

Plateformes... pas seulement mobiles

Un des objectifs de Qt est de supporter une pléthore de plateformes, cependant avec un code utilisateur commun, tant pour les plateformes mobiles que desktop. Côté Qt Quick, on peut atteindre cet objectif par les Qt Quick Components (aux exceptions des spécificités de chaque plateforme), aussi disponibles en édition Desktop. Tout ne sera pas orienté vers les plateformes mobiles, le desktop sera toujours supporté ; de même, les widgets resteront « une option viable ».

Qt 5.0

Il s'agit du projet de recherche et développement le plus grand mené par Digia pour le moment ; jusqu'à présent, une alpha et la première bêta sont disponibles pour recueillir l'avis des utilisateurs : grâce à eux, cette version majeure est en cours de finalisation pour une sortie prévue au dernier trimestre de 2012.

Commentez la news de Thibaut Cuvelier en ligne : Lien [64](#)

La Qt Developer Conference européenne se déroulera du 12 au 14 novembre, le planning des formations est disponible

L'édition européenne de la Qt Developer Conference est maintenant sur les rails. Elle se déroulera du 12 au 14 novembre, à Berlin. On y retrouvera une série de keynotes, de formations, de conférences et bien d'autres.



La date limite pour se présenter comme conférencier étant hier, on devrait avoir sous peu le détail de ce qu'on pourra y découvrir. On sait d'ores et déjà qu'on pourra voir des « représentants talentueux de l'écosystème Qt » pendant ces deux jours.

Comme pour les DevDays, des formations seront possibles le premier jour, mélangeant des présentations *ex cathedra* et des séances pratiques (il est d'ailleurs recommandé de prendre son portable). Les possibilités sont :

- Introduction to Qt for the desktop
- Introduction to Qt Quick
- Introduction to Qt for embedded Linux
- Introduction to Testing Qt application with Squish
- Model/View programming using Qt
- Multithreading with Qt
- Modern OpenGL with Qt5
- Whats new in C++11 (with a Qt5 focus)
- Getting up to speed with git

Il n'y aura pas de foire d'emploi à cause de problèmes de vie privée relevés par bon nombre de potentiels participants, elle sera remplacée par des opportunités de réseautage social.

Un salon d'exposition sera également ouvert pour les trois jours.

Le site de la conférence : Lien [65](#)

Commentez la news de Thibaut Cuvelier en ligne : Lien [66](#)

Déboguer avec OpenGL 4

Lorsque l'on débute l'apprentissage d'OpenGL et des shaders (et même ensuite), on est vite confronté au problème du débogage, soit parce que le programme s'arrête brusquement, soit parce que le résultat obtenu ne correspond pas à ce que l'on attend. Traditionnellement, on utilise la fonction *glGetError*, mais elle est encore trop souvent « oubliée » par les développeurs et elle donne finalement assez peu d'informations.

Heureusement, cette problématique a été prise en compte dans les dernières spécifications d'OpenGL avec l'ajout de nouvelles extensions pour le débogage. Ce billet de blog aborde les fonctionnalités de débogage introduites dans OpenGL 4.1 avec l'extension ARB_debug_output et complétées dans OpenGL 4.3 avec l'extension KHR_debug.

Les outils externes de débogage ne sont pas abordés.

Les nouvelles extensions

L'extension ARB_debug_output (GL 4.1)

Dans la version OpenGL 4.1, a été ajoutée l'extension ARB_debug_output, qui est une promotion de l'extension GL_AMD_debug_output proposée par AMD. Cette extension ajoute la possibilité de créer des contextes avec un mode de débogage. En l'activant, un système d'événements est activé et permet d'obtenir des informations variées, par exemple sur les problèmes rencontrés lors de la compilation des shaders GLSL, des appels de fonction non valides ou des problèmes potentiels de performance. Les messages générés par ce système peuvent être stockés dans une pile (« message log ») ou être récupérés par une fonction callback définie par l'utilisateur.

L'extension KHR_debug (GL 4.3)

Dans le but d'uniformiser les différentes versions d'OpenGL (en particulier avec OpenGL ES), l'évolution d'OpenGL tend à reprendre des fonctions provenant d'OpenGL ES. C'est le cas de l'extension KHR_debug dans OpenGL 4.3 qui est une promotion de l'extension ARB_debug_output d'OpenGL 4.1 et des extensions EXT_debug_marker et EXT_debug_label d'OpenGL ES. Cette extension ajoute ainsi des fonctions pour annoter les événements ou des groupes de commandes OpenGL.

Création d'un contexte de débogage (GL 4.1)

La première chose à faire pour utiliser un contexte de débogage est de vérifier que l'extension ARB_debug_output est bien prise en charge. Un exemple de code pour vérifier une extension est donné dans le tutoriel Les extensions OpenGL : Lien [67](#).

Pour créer un contexte de débogage, il faut simplement créer un contexte en appelant la fonction *CreateContextAttribs* avec le paramètre CONTEXT_DEBUG_BIT. En fonction du système d'exploitation, on aura donc un code équivalent à celui-ci :

```
int attribs[] =
{
#ifdef WIN32
    WGL_CONTEXT_MAJOR_VERSION_ARB, 4,
    WGL_CONTEXT_MINOR_VERSION_ARB, 1,
    WGL_CONTEXT_FLAGS_ARB,
    WGL_CONTEXT_DEBUG_BIT_ARB,
    WGL_CONTEXT_PROFILE_MASK,
    WGL_CONTEXT_CORE_PROFILE_BIT_ARB,
#endif
#ifdef __linux__
    GLX_CONTEXT_MAJOR_VERSION_ARB, 4,
    GLX_CONTEXT_MINOR_VERSION_ARB, 1,
    GLX_CONTEXT_FLAGS_ARB,
    GLX_CONTEXT_DEBUG_BIT_ARB,
    GLX_CONTEXT_PROFILE_MASK,
    GLX_CONTEXT_CORE_PROFILE_BIT_ARB,
#endif
    0
};

// initialisation
HDC hdc; // Handle to a device context
HGLRC hglrc; // Handle to an OpenGL rendering context
hglrc = wglCreateContext (hdc);
wglMakeCurrent (hdc, hglrc);

// contexte attribs
int pixelFormat, numFormats;
glChoosePixelFormatARB(hdc, attribList, NULL, 1,
&pixelFormat, &numFormats);
HGLRC wglCreateContextAttribsARB(HDC hdc, HGLRC
hshareContext, const int *attribList);

// après utilisation
wglMakeCurrent (NULL, NULL);
wglDeleteContext (hglrc);
```

Les fonctionnalités de débogage sont activées en utilisant la constante GL_DEBUG_OUTPUT. Celle-ci peut être activée ou désactivée avec *glEnable* et *glDisable*. Par défaut, cette constante est activée dans un contexte de débogage et désactivée dans le cas contraire.

```
// active les messages de débogage
glEnable(GL_DEBUG_OUTPUT);

// désactive les messages de débogage
glDisable(GL_DEBUG_OUTPUT);
```

Plusieurs niveaux de débogage sont possibles en fonction de la création ou non d'un contexte de débogage et de si l'on active GL_DEBUG_OUTPUT :

- si on ne crée pas un contexte de débogage et que l'on active GL_DEBUG_OUTPUT, les messages envoyés et le contenu sont laissés à l'appréciation des implémentations d'OpenGL, mais l'appel des fonctions de débogage ne produit pas d'erreur ;
- si on crée un contexte de débogage et que l'on

n'active pas `GL_DEBUG_OUTPUT`, aucun message de débogage n'est lancé ;

- si on crée un contexte de débogage et que l'on active `GL_DEBUG_OUTPUT`, toutes les fonctionnalités de débogage sont activées.

Créer des contextes de débogage à partir d'une bibliothèque (GL 4.1)

Il existe plusieurs bibliothèques graphiques qui permettent de travailler sur des contextes OpenGL (SFML, SDL, Qt, etc.). Voyons comment créer un contexte de débogage dans ces cas d'utilisation.

Avec FreeGlut (Lien 68)

FreeGlut fournit directement une constante pour créer un contexte en mode débogage : `GLUT_DEBUG`.

```
// FreeGlut
glutInitContextFlags(GLUT_DEBUG);
```

Avec Qt (Lien 69)

La bibliothèque Qt fournit des outils pour créer et manipuler directement des fenêtres contenant un contexte OpenGL avec le module `QtOpenGL` (Lien 70). Pour créer un contexte spécifique d'une version OpenGL, il suffit d'utiliser la fonction `setVersion` (Lien 71) de la classe `QGLFormat` (Lien 72) :

```
#include <QtOpenGL>
...
QGLFormat format;
format.setVersion(4, 3);
format.setProfile(QGLFormat::CoreProfile); //
nécessite Qt >= 4.8
format.setSampleBuffers(true);
GLWidget* w = new QGLWidget(format);
```

Pour plus de détails, voir l'article [How to use OpenGL Core Profile with Qt \(Lien 73\)](#) sur le wiki de Qt.

Avec SFML 2.0 (Lien 74)

La bibliothèque SFML permet de demander lors de la création d'une fenêtre la version du contexte OpenGL à utiliser avec la classe `sf::ContextSettings` (Lien 75). S'il n'est pas possible de créer un contexte avec les paramètres demandés, SFML va créer un contexte le plus proche possible. Il est donc important de vérifier la version de contexte réellement créée.

```
#include <SFML/OpenGL.hpp>
...
sf::ContextSettings settings;
settings.depthBits = 24;
settings.stencilBits = 8;
settings.antiAliasingLevel = 4;
settings.majorVersion = 4;
settings.minorVersion = 3;

sf::Window window(sf::VideoMode(800, 600),
"OpenGL",
sf::Style::Default, settings);

sf::ContextSettings settings =
window.getSettings();
std::cout << "depth bits:" << settings.depthBits
<< std::endl;
std::cout << "stencil bits:" <<
```

```
settings.stencilBits << std::endl;
std::cout << "antiAliasing level:" <<
settings.antiAliasingLevel
<< std::endl;
std::cout << "version:" << settings.majorVersion
<< "."
<< settings.minorVersion << std::endl;
```

Pour plus de détails, voir l'article [Using OpenGL in a SFML window \(Lien 76\)](#) sur le site de SFML.

Avec SDL 1.3 (Lien 77)

La bibliothèque SDL fournit les constantes `SDL_GL_CONTEXT_MAJOR_VERSION` pour définir la version du contexte que l'on souhaite utiliser.

```
SDL_GL_SetAttribute(SDL_GL_CONTEXT_MAJOR_VERSION,
4);
SDL_GL_SetAttribute(SDL_GL_CONTEXT_MINOR_VERSION,
3);
mainwindow = SDL_CreateWindow(PROGRAM_NAME,
SDL_WINDOWPOS_CENTERED,
SDL_WINDOWPOS_CENTERED, 512, 512,
SDL_WINDOW_OPENGL | SDL_WINDOW_SHOWN);
maincontext = SDL_GL_CreateContext(mainwindow);
```

Pour plus de détails, voir l'article [Creating a Cross Platform OpenGL 3.2 Context in SDL](#) sur le site de SFML : [Lien 78](#).

Les modes de synchronisation (GL 4.1)

Par défaut, les appels de fonction OpenGL sont asynchrones, ce qui veut dire que les fonctions peuvent simplement envoyer les commandes aux contextes OpenGL puis passent à l'instruction suivante dans le code. Par exemple, le code suivant peut poser des problèmes puisque rien ne garantit que l'on mesure réellement le temps d'exécution de la commande :

```
boost::timer t;
glDrawElement(...);
double elapsed_time = t.elapsed();
```

Lorsque l'on débogue une application utilisant OpenGL, il peut alors y avoir un décalage entre l'appel d'une fonction et une erreur générée. Pour faciliter le débogage, il est possible de changer le mode de synchronisation pour forcer OpenGL à attendre la fin de la commande avant de passer à l'instruction suivante dans le code. Cette fonctionnalité garantit la synchronisation dans un contexte, mais pas la synchronisation entre plusieurs contextes, qui reste sous la responsabilité de l'application.

```
// active le mode synchrone
glEnable(GL_DEBUG_OUTPUT_SYNCHRONOUS);
```

Il est possible de tester le mode avec `glIsEnabled` (Lien 79) et de le désactiver avec `glDisable` (Lien 80). Activer cette fonctionnalité peut être très coûteux en termes de performance et doit être utilisée uniquement pour le débogage.

La pile des événements (GL 4.1)

La pile des événements, appelée « message log » contient les messages lancés dans un contexte de débogage. Cette pile est de taille limitée, ce qui signifie que si elle est pleine, les événements les plus anciens seront perdus. Il

faut donc la vider régulièrement en lisant les messages. Lorsque l'on utilise plusieurs contextes, chaque contexte possède sa propre pile des événements. La fonction `glGetIntegerv` ([Lien 81](#)) permet d'obtenir des informations sur les capacités et le contenu de la pile.

```
GLint maxMessages, totalMessages, len, maxLen,
lens[10];
GLenum source, type, id, severity,
severities[10];

glGetIntegerv(GL_MAX_DEBUG_LOGGED_MESSAGES_ARB,
&maxMessages);
printf("Nombre de messages maximum que peut
contenir la pile : %d\n", maxMessages);
```

```
glGetIntegerv(GL_DEBUG_LOGGED_MESSAGES_ARB,
&totalMessages);
printf("Nombre de messages contenus actuellement
dans la pile : %d\n", totalMessages);

glGetIntegerv(GL_MAX_DEBUG_MESSAGE_LENGTH_ARB,
&maxLen);
printf("Taille maximale de message : %d\n",
maxLen);

glGetIntegerv(GL_DEBUG_NEXT_LOGGED_MESSAGE_LENGTH
_ARB, &len);
printf("Taille du prochain message : %d\n", len);
```

Retrouvez la suite du billet blog de Guillaume Belz en ligne : [Lien 82](#)

Les dernières news

Sortie des spécifications d'OpenGL 4.3 et OpenGL ES 3.0

Lors de la conférence SIGGRAPH 2012, le groupe Khronos a annoncé la sortie des spécifications d'OpenGL 4.3 et d'OpenGL ES 3.0.

Cette nouvelle version apporte 27 nouvelles extensions, dont les suivantes :

- ajout d'une nouvelle extension nommée *GL_ARB_compute_shader*, permet d'utiliser un nouveau type de shader : les **Compute Shaders**. Ceux-ci permettent de réaliser des calculs complexes sur les images ou les volumes en bénéficiant de la puissance du parallélisme des cartes graphiques. Ils ont donc la même utilisation que ce que fournit OpenCL. La différence vient du fait que les *Compute Shaders* sont destinés aux calculs s'intégrant dans le pipeline graphique, alors qu'OpenCL est destiné aux calculs ne nécessitant pas le pipeline graphique ;
- ajout d'un nouveau type de *Buffer Object* : les **shader storage buffer objects** (extension *GL_ARB_shader_storage_buffer_object*). Ce sont des espaces mémoire accessibles en écriture et en lecture par tous les types de shaders. Ils facilitent donc la transmission de gros volumes d'informations entre les shaders ;
- les *texture parameter queries* permettent d'obtenir des informations sur le support des textures sur la plateforme d'exécution ;
- les compressions de texture haute qualité **ETC2 / EAC** (extension *GL_ARB_ES3_compatibility*) ;
- l'extension *GL_ARB_arrays_of_arrays* ajoute les tableaux multidimensionnels dans le GLSL (pour pouvoir écrire par exemple `float f[4]`

[3]);

- fonctionnalités de débogage pendant l'exécution ;
- *texture views* pour travailler sur les textures selon différentes manières, sans devoir les dupliquer ;
- les *indirect multi-draws* permettent de réaliser des instanciations multiples en deux temps, le premier temps permettant de faire des calculs et de stocker les paramètres dans un Buffer Object, puis de les réutiliser dans une seconde étape de rendu ;
- amélioration de l'utilisation d'OpenGL par plusieurs applications : sécurisation des espaces mémoire pour éviter qu'une application écrive dans l'espace mémoire d'une autre application (extension *GL_ARB_robust_buffer_access_behavior*) et éviter que le reset de la carte graphique déclenché par une application perturbe les autres applications.

NVIDIA a sorti les drivers 305.53 pour Windows et 304.15.00.02 pour Linux supportant OpenGL 4.3. Télécharger ici : [Lien 83](#).

Quelles sont les nouvelles fonctionnalités qui vous intéressent le plus ?

Que pensez-vous de l'ajout des Compute Shaders, en particulier font-ils double-emploi avec OpenCL ?

Spécifications :

- 3D : OpenGL Core profile 4.3 (06/08/2011) : [Lien 84](#), Compatibility profile 4.3 (06/08/2011) : [Lien 85](#) et OpenGL Shading Language 4.30.6 (06/08/2012) : [Lien 86](#)
- 3D embarqué : OpenGL ES 3.0.0 (06/08/2012) : [Lien 87](#) et OpenGL ES Shading Language 3.00.3 (06/08/2012) : [Lien 88](#)

Commentez la news de Guillaume Belz en ligne : [Lien 89](#)

Binding XML - Java

À travers un exemple concret, cet article présente l'API JAXB qui nous permet de transformer un fichier XML en objets Java, et vice versa.

1. Objectif

Il sera d'obtenir une requête SQL à partir de son nom et du groupe auquel elle appartient. Voyons pour commencer à quoi pourrait ressembler l'interface de notre QueryLoader :

```
interface IQueryLoader
public interface IQueryLoader {
    Map<String, String> getQueries(String
queriesName);
    String getQuery(String groupName, String
queryName);
}
```

Créons ensuite notre fichier XML qui nous servira d'exemple :

Fichier exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<queries>
  <group name="group1">
    <query name="query1">
      SELECT field1 FROM table1
    </query>
    <query name="query2">
      SELECT field2 FROM table2
    </query>
  </group>
  <group name="group2">
    <query name="query3">
      SELECT field3 FROM table3
    </query>
    <query name="query4">
      SELECT field4 FROM table4
    </query>
  </group>
</queries>
```

À partir de ce fichier, nous pouvons écrire assez facilement nos classes Java correspondantes. Tout d'abord notre requête :

```
class Query
public class Query {

    private String name;
    private String query;

    public final String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

```
public final String getQuery() {
    return query;
}

public final void setQuery(String query) {
    this.query = query;
}
}
```

Puis notre groupe de requêtes :

```
class QueryGroup
public class QueryGroup {

    private String name;

    private List<Query> queries = new
LinkedList<Query>();

    public String getName() {
        return name;
    }

    public final void setName(String name) {
        this.name = name;
    }

    public final List<Query> getQueries() {
        return queries;
    }

    public final void setQueries(List<Query>
queries) {
        this.queries = queries;
    }
}
```

Il nous faut aussi une classe pour contenir la liste des groupes de requêtes :

```
class Queries
public class Queries {

    private List<QueryGroup> queryGroups = new
LinkedList<QueryGroup>();

    public final void
setQueryGroups(List<QueryGroup> queryGroups) {
        this.queryGroups = queryGroups;
    }

    public final List<QueryGroup>
getQueryGroups() {
        return queryGroups;
    }
}
```

```
}  
}
```

Il nous reste à voir l'implémentation de notre interface IQueryLoader, nous le ferons très bientôt. Occupons-nous maintenant de JAXB.

2. Présentation de JAXB

JAXB est l'acronyme de Java Architecture for XML Binding ([Lien 90](#)). JAXB est donc capable de sérialiser des objets Java en un fichier XML, et de les désérialiser. Il existe également d'autres API pour manipuler le XML en Java, comme SAX (Simple API for XML : [Lien 91](#)) ou JAXP (Java API for XML Processing : [Lien 92](#)). SAX fait d'ailleurs partie de JAXP. Mais ces API n'ont pas la même finalité que JAXB, elles servent davantage à manipuler et parcourir l'arborescence XML qu'à réaliser la sérialisation d'objets en XML.

Si on prend l'exemple de SAX, on va parcourir l'arborescence XML avec un parser, instancié depuis une factory. Ce parser lit ses données à partir d'un InputStream, et appelle diverses méthodes d'un handler, que nous créons en héritant d'un DefaultHandler ([Lien 93](#)). Le début de l'analyse du document se fait ainsi par l'appel de la méthode startDocument(), l'arrivée sur une balise avec la méthode startElement(), et ainsi de suite pour chaque élément trouvé. Nous devons donc définir le comportement attendu en fonction des cas : est-ce une balise, est-ce celle attendue, ses attributs sont-ils corrects, etc. Tout ceci est plutôt lourd à mettre en place, d'autant plus que notre handler ne conserve aucune donnée en mémoire quand il parcourt le flux XML. C'est à notre implémentation de tout enregistrer.

Si on connaît exactement le format des données attendues, ou mieux, si on dispose du schéma, JAXB est beaucoup plus simple à utiliser, puisque quelques classes Java annotées lui suffisent pour analyser le flux XML. Une API qui se rapprocherait de JAXB est Castor ([Lien 94](#)), qui fait la même chose : transformer un flux XML en POJO, et réciproquement. Mais plus ancienne, elle passe par l'analyse d'un fichier XML de mapping, parfois complexe à écrire.

Il existe d'autres solutions permettant la sérialisation, par exemple les classes XMLEncoder ([Lien 95](#)) et XMLDecoder ([Lien 96](#)), Jakarta Commons Digester ([Lien 97](#)) de la fondation Apache, ou encore l'API XStream. Ne connaissant pas ces solutions, je n'en parlerai pas, et vous laisse les découvrir à travers les tutoriels sur Developpez.com, dont vous trouverez les liens en fin d'article.

3. Utilisation de JAXB

Pour disposer de JAXB dans notre classpath, avec Maven il suffit d'ajouter les dépendances suivantes dans notre pom :

```
Dependency maven pour JAXB  
<dependency>  
  <groupId>javax.xml.bind</groupId>  
  <artifactId>jaxb-api</artifactId>  
  <version>2.2.4</version>
```

```
</dependency>
```

Sinon, on peut le télécharger ici par exemple : [Lien 98](#).

3.1. Annotations de nos classes

Nous avons dit que tout se faisait par annotations. Nous allons commencer par la classe correspondant à la racine de notre arborescence XML. Il s'agit ici de Queries, qui sera annotée avec javax.xml.bind.annotation.XmlRootElement :

```
@XmlRootElement  
public class Queries {  
  ?  
}
```

Cet élément correspond à la balise racine <queries/>, qui est unique par définition. C'est toujours la première balise rencontrée, il est inutile d'indiquer son nom. Il n'en va pas de même de la balise suivante, <group/>, dont nous devons indiquer le nom. L'annotation est javax.xml.bind.annotation.XmlElement, et se place sur le getter :

```
@XmlElement(name = "group")  
public final List<QueryGroup> getQueryGroups() {  
  return queryGroups;  
}
```

Passons maintenant au contenu de cette balise <queries/>, représenté par la classe QueryGroup. Cette balise contient un attribut, son nom. L'annotation est javax.xml.bind.annotation.XmlAttribute :

```
@XmlAttribute  
public String getName() {  
  return name;  
}
```

On retrouve l'annotation @XmlElement pour la balise <Query/> :

```
@XmlElement(name = "query")  
public final List<Query> getQueries() {  
  return queries;  
}
```

Finissons par la classe Query. Elle possède un attribut XML, son nom :

```
@XmlAttribute(name = "name")  
public final String getName() {  
  return name;  
}
```

Et aussi une valeur, la requête SQL, annotée avec javax.xml.bind.annotation.XmlValue :

```
@XmlValue  
public final String getQuery() {  
  return query;  
}
```

Voilà pour ce qui est des annotations. Il n'y a rien de plus à

faire, JAXB est maintenant capable très simplement de faire le lien entre nos POJO et notre fichier XML. La transformation d'objets Java en flux XML se nomme le marshalling, l'unmarshalling étant la transformation de XML en POJO. Et ceci se fait simplement avec une seule ligne de code :

```
Queries queries = JAXB.unmarshal(xmlInputStream,
Queries.class);
```

Et c'est tout. Il ne nous reste plus qu'à tester que tout fonctionne.

3.2. Tests

On écrit une petite classe de test unitaire pour JUnit. Pour tester l'unmarshalling, on part du fichier, et on vérifie que toutes les requêtes sont bien présentes. Pour le test du marshalling, on commence par notre objet « unmarshallé », et qu'on « marshallera » dans une String. Et on « unmarshalle » à nouveau cette String dans un objet Queries, qu'il nous reste à comparer au premier. Voici le code du test :

Classe de test

```
import static org.hamcrest.CoreMatchers.is;
import static org.junit.Assert.assertThat;
// Autres import
public class QueriesTest {

    private Queries queries;

    @Before
    public void before() {
        InputStream xmlStream =
Queries.class.getResourceAsStream("queriesTest.xml");
        queries = JAXB.unmarshal(xmlStream,
Queries.class);
    }

    @Test
    public void unmarshallingTest() throws
Exception {
        List<QueryGroup> queryGroups =
queries.getQueryGroups();

assertThat(queries.getQueryGroups().size(),
is(2));

        QueryGroup group = queryGroups.get(0);
assertThat(group.getName(),
is("group1"));
assertThat(group.getQueries().size(),
is(2));

        List<Query> queryList =
group.getQueries();
        Query query = queryList.get(0);
assertThat(query.getName(),
is("query1"));
assertThat(query.getQuery().trim(),
is("SELECT field1 FROM table1"));

        query = queryList.get(1);
assertThat(query.getName(),
is("query2"));
assertThat(query.getQuery().trim(),
is("SELECT field2 FROM table2"));
```

```
        group = queryGroups.get(1);
assertThat(group.getName(),
is("group2"));
assertThat(group.getQueries().size(),
is(2));

        queryList = group.getQueries();
        query = queryList.get(0);
assertThat(query.getName(),
is("query3"));
assertThat(query.getQuery().trim(),
is("SELECT field3 FROM table3"));

        query = queryList.get(1);
assertThat(query.getName(),
is("query4"));
assertThat(query.getQuery().trim(),
is("SELECT field4 FROM table4"));
    }

    @Test
    public void marshallingTest() throws
Exception {
        StringWriter writer = new StringWriter();
        JAXB.marshal(queries, writer);

        String xmlString = writer.toString();
        System.out.println(xmlString);
        Queries queries2 = JAXB.unmarshal(new
StringReader(xmlString), Queries.class);

        checkQueries(queries2, queries);
    }

    private void checkQueries(Queries queries,
Queries expected) {
        List<QueryGroup> groups =
queries.getQueryGroups();
        List<QueryGroup> expectedGroups =
expected.getQueryGroups();
        checkGroups(groups, expectedGroups);
    }

    private void checkGroups(List<QueryGroup>
groups, List<QueryGroup> expectedGroups) {
        assertThat(groups.size(),
is(expectedGroups.size()));
        Iterator<QueryGroup> iterator1 =
groups.iterator();
        Iterator<QueryGroup> iterator2 =
expectedGroups.iterator();
        while (iterator1.hasNext()) {
            checkGroup(iterator1.next(),
iterator2.next());
        }
    }

    private void checkGroup(QueryGroup group,
QueryGroup expectedGroup) {
        assertThat(group.getName(),
is(expectedGroup.getName()));
        List<Query> queriesList =
group.getQueries();
        List<Query> expectedQueriesList =
expectedGroup.getQueries();
        checkQueriesList(queriesList,
expectedQueriesList);
    }

    private void checkQueriesList(List<Query>
queriesList, List<Query> expectedQueriesList) {
```

```

        assertThat(queriesList.size(),
            is(expectedQueriesList.size()));
        Iterator<Query> iterator1 =
            queriesList.iterator();
        Iterator<Query> iterator2 =
            expectedQueriesList.iterator();
        while (iterator1.hasNext()) {
            checkQuery(iterator1.next(),
                iterator2.next());
        }
    }

    private void checkQuery(Query query, Query
        expectedQuery) {
        assertThat(query.getName(),
            is(expectedQuery.getName()));
        assertThat(query.getQuery().trim(),
            is(expectedQuery.getQuery().trim()));
    }
}

```

Nous avons imprimé notre String XML, ce qui nous permet de vérifier visuellement à quoi elle ressemble. Et maintenant que tout est correct, il nous reste à implémenter notre QueryLoader.

3.3. Implémentation du QueryLoader

Prenons de bonnes habitudes, et commençons par notre classe de test :

```

class XmlQueryLoaderTest
import static org.hamcrest.CoreMatchers.is;
import static org.junit.Assert.assertThat;
// Autres imports
public class XmlQueryLoaderTest {

    private XmlQueryLoader queryLoader;

    @Before
    public void before() {
        InputStream xmlStream =
            XmlQueryLoader.class.getResourceAsStream("queries
            Test.xml");
        queryLoader = new
            XmlQueryLoader(xmlStream);
    }

    @Test
    public void getQueryTest() throws Exception {
        assertThat(queryLoader.getQuery("group1",
            "query1"), is("SELECT field1 FROM table1"));
    }

    @Test
    public void getQueriesTest() throws Exception
    {
        Map<String, String> queries =
            queryLoader.getQueries("group2");
        assertThat(queries.size(), is(2));
        assertThat(queries.get("query3"),
            is("SELECT field3 FROM table3"));
        assertThat(queries.get("query4"),
            is("SELECT field4 FROM table4"));
    }
}

```

Et enfin, l'implémentation à tester :

class XmlQueryLoader

```

public class XmlQueryLoader implements
    IQueryLoader {

    private final Map<String, Map<String,
        String>> queriesGroup = new LinkedHashMap<String,
        Map<String, String>>();

    public XmlQueryLoader(InputStream xmlStream)
    {
        Queries queries =
            JAXB.unmarshal(xmlStream, Queries.class);
        for (QueryGroup group :
            queries.getQueryGroups()) {
            String groupName = group.getName();
            Map<String, String> queryGroup = new
                LinkedHashMap<String, String>();
            queriesGroup.put(groupName,
                queryGroup);
            for (Query query :
                group.getQueries()) {
                String queryName =
                    query.getName();
                String sqlQuery =
                    query.getQuery();
                queryGroup.put(queryName,
                    sqlQuery.trim());
            }
        }
    }

    @Override
    public Map<String, String> getQueries(String
        groupName) {
        return queriesGroup.get(groupName);
    }

    @Override
    public String getQuery(String groupName,
        String queryName) {
        return
            queriesGroup.get(groupName).get(queryName);
    }
}

```

Nous avons utilisé une LinkedHashMap ([Lien 99](#)), car elle nous permet de parcourir les requêtes SQL dans leur ordre de déclaration si on veut les exécuter les unes à la suite des autres, par exemple pour enchaîner la création de tables.

4. Schéma XML

4.1. Génération du schéma

JAXB offre la fonctionnalité de générer un schéma XML, ce qui nous permettra de valider le flux XML reçu. Pour ceci, les quelques lignes de code suivantes suffisent :

Génération du schéma

```

final File baseDir = new File(".");
class MySchemaOutputResolver extends
    SchemaOutputResolver {

    @Override
    public Result createOutput(String
        namespaceUri, String suggestedFileName) throws
        IOException {
        return new StreamResult(new File(baseDir,

```

```
suggestedFileName));
    }
}

JAXBContext context =
JAXBContext.newInstance(Queries.class);
context.generateSchema(new
MySchemaOutputResolver());
```

Nous obtenons un fichier schema1.xsd :

Schéma

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?>
<xs:schema version="1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:element name="queries" type="queries"/>

    <xs:complexType name="queries">
        <xs:sequence>
            <xs:element name="group"
type="queryGroup" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="queryGroup">
        <xs:sequence>
            <xs:element name="query" type="query"
minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="name"
type="xs:string"/>
    </xs:complexType>

    <xs:complexType name="query">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="name"
type="xs:string"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:schema>
```

4.2. Validation du flux

Nous pouvons utiliser le schéma pour valider le flux XML :

Génération avec validation

```
SchemaFactory schemaFactory =
SchemaFactory.newInstance("http://www.w3.org/2001
/XMLSchema");
Schema schema = schemaFactory.newSchema(new
File(baseDir, "schema1.xsd"));

Unmarshaller unmarshaller =
```

```
context.createUnmarshaller();
unmarshaller.setSchema(schema);
Queries unmarshall = (Queries)
unmarshaller.unmarshal(xmlStream);
```

En cas de non-conformité du flux XML par rapport au schéma, la méthode unmarshall() lève une javax.xml.bind.UnmarshalException.

De la même manière que nous avons obtenu notre Unmarshaller, on obtient un Marshaller, auquel on peut préciser le schéma, et qui transformera nos objets en flux XML. Nemeck me fait remarquer que la sérialisation (le marshalling) ne produit pas nécessairement un XML valide vis-à-vis du XSD, et qu'il est donc conseillé de préciser le schéma.

4.3. Génération des classes

Le schéma ne sert pas qu'à la validation du flux XML. Nous pouvons aussi l'employer pour générer nos classes, à l'aide de l'utilitaire xjc du JDK :

```
xjc schema.xsd
```

Les classes générées sont par défaut dans le package generated. On retrouve bien nos trois classes Queries, QueryGroup et Query, ainsi qu'une classe ObjectFactory. Je vous laisse découvrir à quoi ressemblent ces classes, elles sont très proches des nôtres, avec essentiellement quelques commentaires Javadoc en plus.

Les deux options les plus utiles de xjc sont :

- help : affiche l'aide et les différentes options disponibles ;
- p : permet de préciser le package. Par exemple avec -p fr.atatorus.gen, les classes seront dans fr/atatorus/gen et non plus dans generated ;
- d : précise le répertoire où seront les classes générées.

5. Conclusion

Voilà, j'espère que ce petit tutoriel vous a permis de découvrir JAXB, et d'apprécier sa facilité d'utilisation. Pour ma part, j'ai vraiment aimé sa simplicité. Ma première expérience avec le binding Java XML a commencé avec JAXP, où on devait tout faire soi-même. La découverte de Castor a été un soulagement, même si j'ai dû me débattre avec les fichiers de mapping. À côté, JAXB est un vrai bonheur !

Si vous voulez découvrir toutes les possibilités de JAXB, je vous renvoie à sa documentation : Lien [100](#).

Retrouvez l'article de Denis Thomas en ligne : [Lien 101](#)

JavaOne 2012 démarre en force : innovation, Cloud, GPU, mobile, résumé de la 1re journée de la plus grande conférence autour de Java

JavaOne 2012, la grand-messe annuelle des développeurs et experts de l'industrie autour de l'écosystème Java a ouvert ses portes hier.

Pendant cinq jours, le Masonic Auditorium de San Francisco sera le théâtre de plus 500 sessions présentées par près de 540 conférenciers, autour du thème central « préparer Java du futur ».

Strategy Keynote

La conférence s'est ouverte avec une session sur la stratégie d'Oracle pour Java. Le tableau de bord de l'éditeur pour l'année 2012 est axé principalement autour de trois domaines : l'innovation technique, la participation communautaire et le leadership d'Oracle.

Au cours de cette session, l'éditeur a fait un résumé des évolutions en 2012 dans Java 7, JavaFx, JEE 8, JDK 8 et 9, etc. Les points évoqués ont été les sorties rapides des mises à jour pour Java 7 ainsi que la forte migration vers cette version.

Des détails ont été dévoilés sur la feuille de route de Java, JDK 8 et 9. La caractéristique de JDK 8 mise en avant est le projet Nashorn, un moteur d'exécution JavaScript entièrement développé en Java par Oracle. Il est basé sur la machine virtuelle Da Vinci et profite d'invokedynamic pour fournir des performances élevées. Le projet Nashorn bénéficie déjà du soutien d'IBM, RedHat et Twitter. JDK 8 sera disponible dans NetBeans 7.3.

En ce qui concerne JavaFX, Oracle a annoncé que sa plateforme de création d'applications internet riches (RIA) était désormais disponible pour toutes les plateformes majeures, y compris Linux ARM. L'éditeur a rappelé que l'outil était livré avec la version actuelle de Java et que NetBeans 7.2 intégrait SceneBuilder, un outil pour la création d'interfaces graphiques avec JavaFX.

Sortie de Java Embedded Suite 7.0 et Oracle Java ME Embedded 3.2.

Nandini Ramani, vice-présidente de la division développeur chez Oracle a annoncé la sortie d'Oracle Java Embedded Suite 7.0 et Oracle Java ME Embedded 3.2 ([Lien 102](#)). Oracle Java Embedded Suite est une plateforme de développement qui facilite la création des applications pouvant s'exécuter à travers une large gamme de systèmes embarqués, tandis que Java ME Embedded 3.2 est un environnement complet d'exécution Java optimisé pour les microcontrôleurs et autres dispositifs limités en ressources.

IBM Keynote

Les présentations d'IBM, l'unique Diamond Sponsor de l'événement, ont directement suivi la Strategy Keynote d'Oracle. Les conférenciers de la société ont essentiellement évoqué le support du Cloud au sein du langage.

IBM s'est également penché sur le partage des ressources (Sharing), avec notamment le Cache pour partager les classes compilées par le JIT, le Multi-tenancy pour l'adaptation dynamique de plusieurs instances d'exécution du JVM aux changements suivant la disponibilité des ressources ainsi que l'isolement dans une seule JVM.

Enfin, IBM a présenté son hardware System Z, spécialement optimisé pour l'exécution de Java ainsi que l'impact important du hardware sur les performances.

Application mobile Scala1

Typesafe, la société fondée par le créateur du langage de programmation moderne Scala a dévoilé l'application mobile Scala1. Scala1 permet aux développeurs de trouver facilement de nouvelles discussions sur Scala, de collaborer avec les autres développeurs, de s'informer sur les évolutions du langage, de trouver rapidement des experts Scala, etc.

L'application open source Scala1 est disponible sur Android et iOS et son code source est publié sur GitHub.

Accélération GPU pour les applications Java

JavaOne a été l'occasion pour AMD de présenter le projet Sumatra, une modification de la JVM pour utiliser le GPU de manière transparente pour le programmeur.

Le projet Sumatra vise à mettre fin à l'utilisation des bibliothèques externes pour l'accélération GPU des applications Java ainsi qu'au processus de conversion entre Java et OpenCL (bibliothèque multiplateforme qui permet d'utiliser les processeurs graphiques pour réaliser des calculs lourds).

L'idée autour du projet est de profiter des structures de données dans la mise en œuvre des outils OpenJDK et de laisser la machine virtuelle Java générer et compiler le code OpenCL en fonction des indices dans le code. Le projet Sumatra pourrait sortir au même moment que Java 9.

JavaOne 2012 s'achèvera le 4 octobre et d'ici là, Oracle va encore dévoiler plusieurs nouveautés autour de la plateforme Java.

Commentez la news d'Hinault Romaric en ligne : [Lien 103](#)

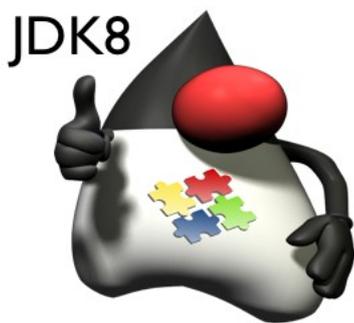
JavaOne 2012 : Oracle dévoile ses plans pour Java, les préversions du JDK 8 et JavaFX Scene Builder 1.1 disponibles

JavaOne 2012, la plus grande conférence annuelle autour de la plateforme Java se déroule actuellement à San Francisco.

Oracle a profité de l'occasion pour dévoiler sa feuille de route pour Java et renouveler son engagement à faire évoluer le langage afin de rassurer les utilisateurs sceptiques face aux récentes découvertes de failles dans la plateforme.

Pendant ces prochains mois, Oracle travaillera essentiellement sur Java SE 8. Au menu : l'éditeur prévoit l'intégration des expressions lambda, du moteur JavaScript Nashorn (déjà disponible dans la préversion de NetBeans 7.3 : [Lien 104](#)), les annotations, la nouvelle API « date and time » et bien plus.

Oracle a également officialisé le projet Sumatra, qui vise à mettre fin à l'utilisation des bibliothèques externes pour l'accélération GPU des applications Java ainsi qu'au processus de conversion entre Java et OpenCL (bibliothèque multiplateforme qui permet d'utiliser les processeurs graphiques pour réaliser des calculs lourds). Le projet Sumatra sera disponible avec Java SE 8.



La préversion du JDK 8 peut déjà être testée sur le site java.net. La sortie de la version finale est prévue pour septembre 2013. On va regretter cependant le report à Java 9 du projet Jigsaw, qui devait ajouter au langage un système de module.

Java EE 7 (l'édition pour entreprise) est annoncé pour avril 2013 et apportera comme nouveautés : une API WebSocket, l'API JCACHE pour une mise en cache temporaire des objets Java, un modèle de programmation pour applications Batch et une API pour la manipulation des données au format JSON. Avec cette version de Java EE, Oracle proposera Glassfish 4.0 comme plateforme de référence.

En ce qui concerne sa plateforme de développement d'Applications Internet Riches JavaFX, Oracle envisage de passer de JavaFX 2.0 (la version actuelle) à JavaFX 8 lorsque sortira le JDK 8. Cette version de JavaFX disposera par défaut au niveau de l'interface utilisateur d'une boîte à outils pour Java SE 8 Embedded. Une

nouvelle API sera proposée pour la création d'interfaces utilisateur avec un meilleur support des balises HTML5, WebView et JavaFX Scene Builder 2.0.

JavaFX Scene Builder, l'outil de mise en page visuelle pour la plateforme JavaFX, qui permet aux utilisateurs de créer des interfaces utilisateur en glissant et en positionnant les composants à partir d'une palette dans une scène est actuellement disponible en version 1.1 développeur avec un support pour Linux 32 et 64 bits.

Toutes ces annonces montrent l'engagement d'Oracle à vouloir faire évoluer l'écosystème Java.

Commentez la news d'Hinault Romaric en ligne : [Lien 105](#)

JavaOne 2012 : Oracle présente la spécification JSR 353, l'API Java pour la manipulation avec souplesse du format JSON

JavaOne 2012 s'est achevé hier. L'événement Java le plus important de l'année a levé le voile sur un nombre impressionnant de nouveautés, innovations et ambitions pour l'écosystème Java.

Oracle pendant ses sessions a présenté sa feuille de route ([Lien 106](#)) pour le langage et les points sur lesquels l'entreprise travaille actuellement pour la prochaine version de Java, dont l'intégration des expressions lambda, du moteur JavaScript Nashorn, des annotations, de la nouvelle API « date and time » et bien plus.

Une session a été entièrement réservée au support du format JSON dans les futures versions du langage. Pour rappel, JSON (JavaScript Object Notation) est un format de données textuel, générique, dérivé de la notation des objets du langage ECMAScript. Il permet de représenter de façon structurée des informations.

L'avantage de ce format de données est qu'il est léger, flexible et facilement interprétable pour les machines. JSON a été adopté comme format d'échange par plusieurs sites Web populaires qui offrent des services Web RESTfull reposant sur le format, notamment Twitter et Amazon.

Actuellement, les développeurs Java utilisent différentes bibliothèques tierces pour produire et consommer des données JSON. Il devient donc nécessaire de normaliser une API JSON afin que les applications qui utilisent le format n'aient plus besoin de regrouper des bibliothèques externes. C'est le but de la spécification JSR 353 qui a été présentée par Oracle à JavaOne 2012.

La spécification JSR 353 : API Java pour le traitement de JSON, vise à rendre l'utilisation de JSON avec le langage plus légère, plus propre et plus cohérente. La spécification JSR 353 décrit actuellement une API (Streaming API) pour produire et consommer en continu des données JSON et une API (Object Model API) du modèle objet pour représenter JSON.

L'API Streaming implémentera les méthodes JsonParser

pour la conversion et JsonGenerator pour la génération. L'API Object Model quant à elle, implémentera les méthodes JsonObject et JsonArray, ainsi que JsonBuilder, JsonReader et JsonWriter.

La spécification est actuellement au stade d'examen précoce et est mise en œuvre en tant que projet open source sur java.net. La fin de la phase d'examen précoce est prévue pour le 7 octobre prochain.

Le groupe d'experts travaillant sur cette spécification inclut les employés d'Oracle, RedHat, Twitter ainsi que des membres autonomes.

Les détails sur la spécification JSR 353 : Lien [107](#)

Commentez la news d'Hinault Romaric en ligne : Lien [108](#)



JavaOne 2012 : Oracle sort la Preview de NetBeans 7.3 et dévoile le projet Easel une extension pour la création des clients RESTful à base de JavaScript

JavaOne 2012 bat son plein. Le Masonic Auditorium de San Francisco vibre aux couleurs de l'écosystème Java qui est en train d'être disséqué par les experts de l'industrie.

Lors de la session consacrée à NetBeans, l'environnement de développement intégré open source pour Java, PHP, C et C++, Oracle a annoncé la sortie de la preview de NetBeans 7.3, la prochaine mise à jour majeure de l'EDI.



Le futur standard du Web HTML5 a volé la vedette à Java durant cette session. La nouveauté phare de NetBeans 7.3 est une meilleure prise en charge du HTML5, CSS3 et JavaScript, permettant de développer plus rapidement des applications Web riches et mobiles.

Les développeurs d'applications Web et mobiles pourront trouver comme nouvelles fonctionnalités dans cette version : un nouvel éditeur de code HTML5 avec la complétion de code pour les éléments HTML5 ; un nouvel éditeur et débogueur JavaScript basé sur le projet Nashorn, la complétion de code pour jQuery, une meilleure prise en charge du CSS3, la génération du code client JavaScript pour les services REST Java existants.

Pour rappel, le projet Nashorn est un moteur d'exécution JavaScript entièrement développé en Java par Oracle. Il est basé sur la machine virtuelle Da Vinci et profite d'invokedynamic pour offrir des performances élevées.

En fournissant un environnement unique où les développeurs peuvent créer des services Java et des clients basés sur HTML5, la préversion de NetBeans 7.3 offre à ceux-ci des outils pour créer et déboguer des applications Web et mobiles qui consomment les services Java en utilisant les derniers standards du Web.

La préversion de NetBeans 7.3 est disponible pour Windows, Mac OS X, Solaris et Oracle Linux.

Toujours concernant NetBeans et HTML5, Oracle a dévoilé le projet Easel, une plateforme qui permet le développement HTML5 et JavaScript dans un environnement Java.

Le projet Easel dispose d'une interface d'édition pour HTML5, JavaScript et jQuery avec support de l'autocomplétion. Il intègre un nouveau débogueur JavaScript basé sur des API de débogage à distance de WebKit.

Le but du projet Easel est de permettre aux développeurs de créer rapidement des clients JavaScript (en utilisant MVC) pour consommer ou tester des services RESTful.

La première version d'Easel pourrait sortir la semaine prochaine comme une extension pour NetBeans. Le support de JDeveloper est prévu pour la version suivante.

Télécharger la préversion NetBeans 7.3 : Lien [109](#)

Commentez la news d'Hinault Romaric en ligne : Lien [110](#)

JFace Databinding avec des composants JFace

Ce tutoriel se propose de détailler l'utilisation du framework JFace Databinding sous Eclipse afin de permettre la liaison entre un modèle de données et des composants de visualisation JFace. Cet article se situe donc dans la droite lignée de l'article JFace Databinding avec SWT ([Lien 111](#)).

1. Introduction

L'API JFace Databinding d'Eclipse permet de lier facilement les données du modèle objet aux informations affichées par l'interface graphique. Il est en effet intéressant que les données entrées par l'utilisateur soient répercutées directement dans le modèle, et vice versa. Dans le précédent article, nous avons étudié les différents mécanismes de l'API au travers d'un exemple simple qui utilisait uniquement des composants graphiques SWT. On se propose ici de mettre en œuvre ce framework sur des composants graphiques JFace. Pour cet article, il est nécessaire d'avoir des connaissances de base dans les domaines suivants :

- développement de plugins avec Eclipse ([Lien 112](#)) : création de vues, gestion des dépendances ;
- utilisation de SWT ([Lien 113](#)), notamment sur les différents composants et leurs propriétés.

2. Un premier exemple

Dans ce premier exemple, nous allons lier de manière la plus simple possible une liste d'éléments de notre modèle à un viewer JFace. Commençons par modifier notre modèle de manière à travailler sur une liste d'objets. Pour cela, nous nous basons sur la classe `PersonBean.java` que nous avons créée dans le premier article. Nous créons ensuite la classe `PersonList` qui peut nous retourner une liste d'objets `PersonBean` grâce à une méthode statique :

PersonBean.java

Voir le code en ligne : [Lien 114](#)

PersonList.java

```
/**
 * Cette classe comprend une unique methode
 * statique qui permet de creer une
 * liste d'objets PersonBean.
 * @author A. Bernard
 */
public class PersonList {

    public static List<PersonBean>
    getPersonList() {
        List<PersonBean> result = new
        ArrayList<PersonBean>();
        result.add(new PersonBean("Bernard", "Alain",
        24, true));
        result.add(new PersonBean("Dupont", "Michel",
        49, true));
        result.add(new PersonBean("Jacques",
        "Jocelyne", 51, false));
        result.add(new PersonBean("Martin",
        "Francois", 31, true));
    }
}
```

```
return result;

    }
}
```

Nous allons maintenant afficher notre liste de personnes dans un viewer JFace de type tableau. Créons la vue « FirstView » au sein de notre application de la manière suivante :

FirstView.java

Voir le code en ligne : [Lien 114](#)

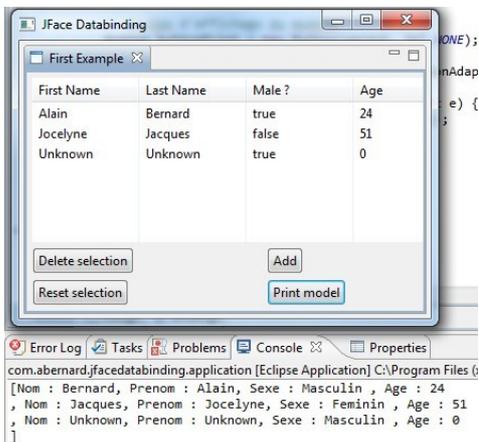
Si l'on observe ce code, on s'aperçoit que le viewer ne possède ni `ContentProvider`, ni `LabelProvider`. Il n'y a pas non plus d'appel effectué à la méthode `setInput`. Au lieu de cela, on utilise un objet de type `WritableList` que l'on initialise grâce à la ligne :

```
input = new
WritableList(PersonList.getPersonList(),
PersonBean.class);
```

D'autre part, on lie le contenu du viewer directement à cette liste grâce à l'instruction :

```
ViewerSupport.bind(viewer, input,
BeanProperties.values(new String[] {"firstname",
"name", "male", "age"}))
```

On indique qu'on lie les valeurs « `firstname` », « `name` », « `male` » et « `age` » aux différentes colonnes du tableau. La classe `ViewerSupport` permet de simplifier la mise en place du databinding pour les viewers JFace. Il permet d'enregistrer les changements effectués au sein du modèle dans sa globalité, mais aussi ceux effectués sur les éléments individuels. C'est cette classe qui se charge de créer automatiquement le `ContentProvider` et le `LabelProvider`. Lançons l'exemple pour constater le fonctionnement du mécanisme :



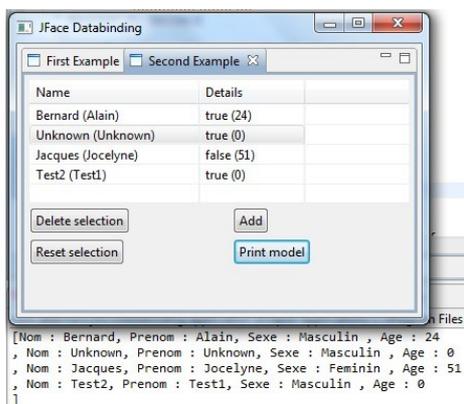
Application et modèle après réinitialisation et suppression d'un élément

3. Observer les changements plus précisément

La première vue que nous avons créée affiche tous les éléments, mais nous laisse peu de latitude quant à la personnalisation de l'affichage en lui-même : le LabelProvider est générique et on ne peut pas le modifier. Grâce aux éléments que nous allons mettre en œuvre, nous allons pouvoir remédier à ce problème. Créons la vue « SecondView » de la manière suivante :

Voir code en ligne : [Lien 115](#)

Nous pouvons visualiser le résultat et vérifier le bon fonctionnement de l'application :



Utilisation des ObservableMap

Il nous faut cette fois-ci définir le ContentProvider. La classe ObservableListContentProvider nécessite que les données d'entrée du viewer implémentent l'interface ObservableList. Notons qu'une liste classique peut être transformée en ObservableList grâce à la classe Properties, comme le montre l'exemple ci-dessous :

```
List<PersonBean> persons =
PersonList.getPersonList();
ObservableList input =
Properties.selfList(PersonBean.class).observe(p
ersons);
```

Afin d'observer les modifications dans les listes d'éléments, on utilise des objets ObservableMap. Ces objets permettent d'observer les modifications d'un attribut au sein des différents éléments d'une collection de type ObservableSet. Cette collection est obtenue directement à

partir du ContentProvider grâce à l'instruction :

```
IObservableSet knownElements =
contentProvider.getKnownElements();
```

On peut ensuite créer un objet ObservableMap pour chaque attribut du modèle pour lequel on souhaite observer les changements. Enfin, on peut créer un LabelProvider pour notre viewer, en passant en paramètre un tableau d'ObservableMap :

```
// Creation du LabelProvider pour le tableau
ILabelProvider labelProvider = new
ObservableMapLabelProvider(labelMaps) {
@Override
public String getColumnText(Object
element, int columnIndex) {
if (columnIndex == 0) {
return names.get(element) + " (" +
firstNames.get(element)
+ ")";
} else {
return genders.get(element) + " (" +
ages.get(element) +
")";
}
}
};
```

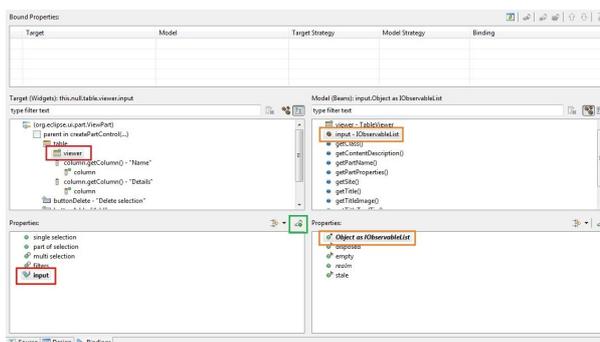
Comme nous avons pu le constater lorsque nous avons exécuté notre exemple, les données sont bien mises à jour en temps réel.

4. Utilisation de WindowBuilder

Une fois encore, WindowBuilder nous permet de réaliser nos bindings facilement. Créons la vue « ThirdView » en l'initialisant de la manière suivante :

Voir le code en ligne : [Lien 116](#)

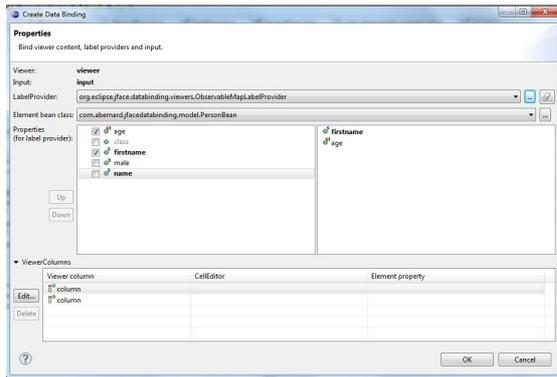
Nous avons initialisé nos données d'entrée au travers de la variable « input », mais nous n'avons pas encore défini le ContentProvider ni le LabelProvider. Ouvrons la vue avec WindowBuilder, et rendons-nous dans l'onglet « Bindings ». Dans la partie gauche, sélectionnons l'objet graphique « viewer » et sa caractéristique « input » (carrés rouges). Dans la partie droite, sélectionnons l'objet « input » et la caractéristique « Object as ObservableList » (carrés orange). Enfin, créons le binding en cliquant sur le bouton idoine (carré vert) :



Première étape avec WindowBuilder

Une fenêtre s'ouvre et nous permet de sélectionner les propriétés à afficher dans les colonnes de notre viewer.

Elle nous permet aussi de définir des éditeurs pour les cellules :



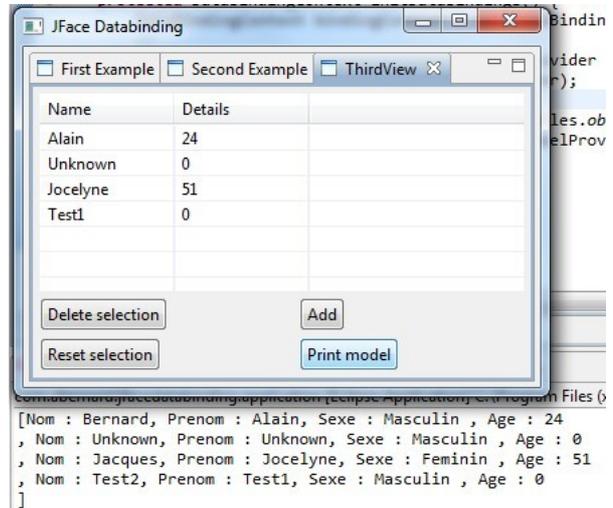
Deuxième étape avec WindowBuilder

Le binding est créé. On retrouve dans le code source généré les éléments que nous avons mentionnés dans cet article :

```
protected DataBindingContext
initDataBindings() {
    DataBindingContext bindingContext = new
DataBindingContext();
    //
    ObservableListContentProvider
listContentProvider = new
ObservableListContentProvider();
viewer.setContentProvider(listContentProvider
);
    //
IObservableMap[] observeMaps =
BeansObservables.observeMaps(listContentPro
vider.getKnownElements(),
PersonBean.class, new String[]{"firstname",
"age"});
viewer.setLabelProvider(new
ObservableMapLabelProvider(observeMaps));
    //
}
```

```
viewer.setInput(input);
//
return bindingContext;
}
```

En lançant l'application, nous pouvons constater le fonctionnement correct du databinding :



Résultat avec WindowBuilder

5. Liens utiles

JFace Databinding sur le site de Lars Vogel : Lien [117](#).

6. Conclusion

Dans ce second article, nous avons vu comment mettre en œuvre le framework JFace Databinding sur des composants JFace. Cela nous permet d'observer les changements effectués sur une collection d'objets, soit de manière très simple et rapide, soit de manière plus précise.

Retrouvez l'article d'Alain Bernard en ligne : Lien [118](#)

Android

Les dernières news



Android est-il un OS de Geeks ?

Il serait fait par des ingénieurs qui travaillent dans leur coin et qui n'écotent jamais personne

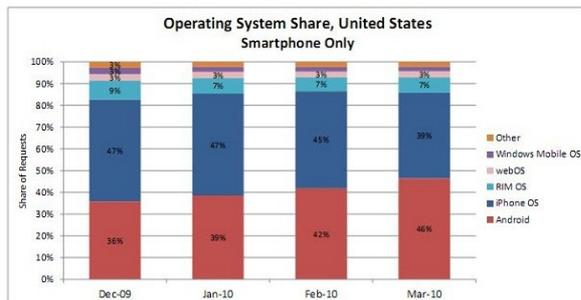
Faites un test simple. Prenez une tablette ou un Smartphone sous Android. Mettez-le dans les mains du premier venu (vos enfants, votre conjoint, vos grands-parents, peu importe). Et observez.

Sauf à ceux qui travaillent dans l'IT ou ont été familiarisés avec l'OS, il y a fort à parier que votre « *cobaye* » soit vite déboussolé.

Recommencez avec un iPad ou un iPhone, et l'avis sera neuf fois sur dix radicalement différent.

Je calme tout de suite les fans boys. Il ne s'agit pas de dire qu'Android est moins bon qu'iOS.

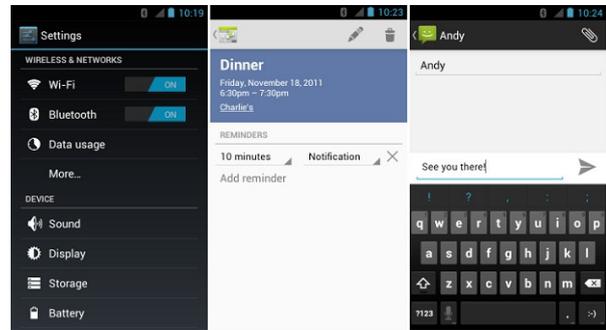
Il s'agit de se demander si Android est adapté au grand public et surtout ce que Google pourrait faire pour améliorer l'expérience utilisateur de son système d'exploitation. Un système par ailleurs numéro un mondial incontesté du secteur.



Android, numéro un mondial du secteur mobile

Prenez un des développeurs d'*Infinite Flight*, le meilleur simulateur de vol disponible sur iOS ([Lien 119](#)) et sur Windows Phone ([Lien 120](#)) avec *X-Plane*.

Assez peu fan d'Apple et possesseur parfaitement comblé d'un Smartphone sous Android, son avis n'en est pas moins tranché sur la question : « *Si aujourd'hui on me demande un conseil pour acheter un Smartphone, sans hésiter je réponds "un iPhone ou un Windows Phone". Android c'est très bien, mais au bout d'un moment tu te rends compte que c'est compliqué. Des fois tu te dis "mais qu'est-ce que c'est que ce bazar ?" Y a des boutons de partout ! Ça part dans tous les sens. Il n'y a pas de consistance (sic)* ».



Son exemple est intéressant pour deux raisons.

La première c'est qu'il n'est absolument pas anti-Android. Au contraire. Mais pour lui, c'est un OS qui perd le grand public. Ou plus exactement que le client achète sans le savoir (« *on achète un Samsung, pas un Android* ») juste parce qu'il veut un téléphone et pas nécessairement un *Smartphone*.

La deuxième, c'est que ce jeune développeur français travaille depuis des années dans la Silicon Valley et qu'il y a lié des relations, notamment avec des employés de Google. Des employés qui ne cachent pas que les équipes marketing peuvent parfois s'arracher les cheveux.

« *Un pote de chez Google m'a expliqué pourquoi [Android est si complexe]. Android est fait par des équipes d'ingénieurs qui travaillent dans leur coin. À chaque fois qu'un nouveau arrive, il rajoute son truc. Et ils n'écotent jamais personne de l'extérieur* », nous expliquait-il.

En résumé, un OS fait par des Geeks, pour des Geeks.

Ce qui en soi n'est pas un mal, mais pose la question de savoir si vous le recommanderiez à vos enfants, votre conjoint, vos grands-parents, etc.

Personnellement, je l'ai fait. Et on ne m'en a pas remercié.

Commentez la news de Gordon Fowler en ligne : [Lien 121](#)

Android : de nouvelles sources de menace à l'horizon

Des chercheurs s'apprentent à publier un framework modulaire d'exploit

Une nouvelle source de menace pour l'OS Android se profile. Deux experts en sécurité affirment qu'ils publieront le mois prochain un nouveau framework open source modulaire appelé AFE (Android Framework for Exploitation).

Cet outil permettra à toute personne bien intentionnée (ou pas du tout !) de créer et de personnaliser son application Android malveillante. Adity Gupta et Subho Halder sont

les deux créateurs derrière cet outil, et se considèrent comme des « *white-hat hackers* », spécialisés dans la sécurité mobile.

Ils présenteront leur Framework dans le ToorCamp, qui se déroulera ce mois-ci dans l'état de Washington, ainsi que dans la NullCon prévue à Delhi le mois prochain.

Les modules raccourcissent considérablement le temps et simplifient le développement d'un logiciel malveillant. Ils permettent à l'utilisateur de choisir parmi 20 fonctionnalités préconstruites, comme la récupération des contacts, des emails, des données de la carte SD, ou l'accès à l'emplacement du mobile en utilisant le dispositif GPS. On peut même forcer la numérotation de n'importe quel numéro premium.

AFE est essentiellement écrit en PHP, Ruby, Bash et Python.

Les deux experts considèrent cet outil comme une option plus dévastatrice que la méthode "classique", qui requiert de prendre « *une application légitime, la décompiler en utilisant APKTool ou Dex2Jar et JF-GUI, insérer nos codes, puis la redistribuer* ». En effet, cet outil permet de masquer le malware en tant qu'application légitime, tels un explorateur de fichier, ou un jeu Tic Tac Toe, pour faciliter sa dissémination dans Google Play.

Après distribution, il ne reste plus qu'à tromper l'utilisateur pour l'amener à télécharger l'application via l'Android

Marketplace, qui permet aux développeurs de publier des applications de façon anonyme.

Mais qu'est-ce qui pousse deux White Hats à mettre une telle plaie entre les mains des pirates en herbe ? Les chercheurs ne l'expliquent pas, en tout cas pas avant le ToorCamp.

Mais on se doute qu'ils fassent partie de ces nombreux chercheurs qui croient que Google pourrait mieux faire pour protéger les utilisateurs de sa plateforme.

Afin de détecter ces menaces, Google a pourtant mis en place en février dernier un mécanisme appelé Bouncer, conçu pour détecter automatiquement les comportements malveillants. Néanmoins, une présentation au Black Hat le mois dernier a permis d'illustrer par quelle facilité ce Bouncer peut être vaincu.

La semaine dernière, Google a aussi mis à jour le guide des développeurs d'applications Android publiées dans le Google Play, afin de contrer le problème des spams, des applications truquées et celles qui violent la vie privée. Les applications existantes seront supprimées en 30 jours, si leurs développeurs ne suivent pas la ligne directrice fixée par Google.

Espérons que ces mesures seront assez suffisantes pour protéger les utilisateurs d'Android.

Commentez cette news de Tarik Zakaria en ligne : [Lien 122](#)

Liens

- Lien 01 : <http://www.w3.org/TR/html5/embedded-content-0.html#dom-innerhtml>
- Lien 02 : http://blog.developpez.com/web/p11267/web/html5_quelques_nouveautes_de_l_api_dom_p
- Lien 03 : <https://developer.mozilla.org/en-US/docs/DOM/window.navigator.onLine>
- Lien 04 : <https://github.com/Progi1984/DVP>
- Lien 05 : <http://f-lefevre.developpez.com/tutoriels/html5/offline/>
- Lien 06 : <http://torgar.developpez.com/articles/css/bordures-css3/>
- Lien 07 : <http://www.pallier.org/ressources/dicofr/dicofr.html>
- Lien 08 : <http://www.pallier.org/ressources/dicofr/liste.de.mots.francais.frgut.txt>
- Lien 09 : <http://sourceforge.net/projects/lazarus/files/>
- Lien 10 : <http://lazarus.developpez.com/cours/mots-croises/>
- Lien 11 : <http://datametric.developpez.com/tutoriels/sas/sas-batch-modes-checkpoint-restart/>
- Lien 12 : <http://cafeine.developpez.com/access/tutoriel/recherchemulti/>
- Lien 13 : <http://jeannot45.developpez.com/articles/access/recherchemulticriteres/>
- Lien 14 : <http://office.microsoft.com/fr-ca/access-help/concepts-de-base-sur-la-conception-d-une-base-de-donnees-HA001224247.aspx>
- Lien 15 : <http://argyronet.developpez.com/office/access/highlightrecord/>
- Lien 16 : <http://argyronet.developpez.com/office/vba/convention/>
- Lien 17 : <http://jeannot45.developpez.com/articles/access/implantationsousetat/>
- Lien 18 : <http://claudeloup.developpez.com/tutoriels/access/comment-implanter-un-sous-formulaire/Comment%20implanter%20un%20sous-formulaire.mdb>
- Lien 19 : <http://claudeloup.developpez.com/tutoriels/access/comment-implanter-un-sous-formulaire/>
- Lien 20 : <http://cpp.developpez.com/gotw>
- Lien 21 : <http://cpp.developpez.com/boost/proto>
- Lien 22 : <http://cpp.developpez.com/carmack>
- Lien 23 : <http://cpp.developpez.com/flottant>
- Lien 24 : <http://cpp.developpez.com/bas-niveau>
- Lien 25 : <http://www.developpez.net/forums/f19/c-cpp/cpp/>
- Lien 26 : <http://cpp.developpez.com/>
- Lien 27 : <http://www.developpez.net/forums/private.php?do=newpm&u=268393>
- Lien 28 : <http://www.developpez.net/forums/d1258293/c-cpp/cpp/programme-rentree-rubrique-cpp/>
- Lien 29 : http://en.wikipedia.org/wiki/IEEE_754
- Lien 30 : http://en.wikipedia.org/wiki/IEEE_754-2008
- Lien 31 : http://en.wikipedia.org/wiki/Aliasing_%28computing%29
- Lien 32 : <http://labs.qt.nokia.com/2011/06/10/type-punning-and-strict-aliasing/>
- Lien 33 : <http://blog.lvm.org/2011/05/what-every-c-programmer-should-know.html>
- Lien 34 : <http://fr.wikipedia.org/wiki/Endianness>
- Lien 35 : <http://cpp.developpez.com/redaction/data/pages/rubrique/cpp/flottant/flottant1/>
- Lien 36 : <http://altdvblogaday.com/2011/10/25/why-i-became-an-educator/>
- Lien 37 : <http://www.udemy.com/cs-107-programming-paradigms/>
- Lien 38 : <http://www.swansontec.com/sregisters.html>
- Lien 39 : <http://www.jegerlehner.ch/intel/>
- Lien 40 : http://en.wikipedia.org/wiki/X86_instruction_listings
- Lien 41 : <http://home.comcast.net/%7Efbui/intel.html>
- Lien 42 : <http://cpp.developpez.com/redaction/data/pages/rubrique/cpp/bas-niveau/bas-niveau1/>
- Lien 43 : <http://www.coverity.com/>
- Lien 44 : <http://msdn.microsoft.com/en-us/library/d3bbz7tz%28v=VS.100%29.aspx>
- Lien 45 : <http://randomascii.wordpress.com/category/code-reliability/>
- Lien 46 : <http://www.viva64.com/en/pvs-studio/>
- Lien 47 : <http://www.viva64.com/en/developers-resources/>
- Lien 48 : <http://www.gimpel.com/html/pcl.htm>
- Lien 49 : http://www.riverblade.co.uk/products/visual_lint/index.html
- Lien 50 : <http://cpp.developpez.com/redaction/data/pages/rubrique/cpp/carmack/analyse-statique/>
- Lien 51 : http://blog.developpez.com/gpu/p10958/langages-frameworks/c/les_signaux_et_slots_dans_qt5
- Lien 52 : http://blog.developpez.com/gpu/p10904/langages-frameworks/opengl/opengl_dans_qt5
- Lien 53 : <http://code.google.com/p/v8/>
- Lien 54 : <http://qt-project.org/wiki/Ot-Platform-Abstraction>
- Lien 55 : <http://blog.developpez.com/gpu/p11283/langages-frameworks/qt/les-modules-de-qt-5/>
- Lien 56 : <http://labs.qt.nokia.com/2011/05/09/thoughts-about-qt-5/>
- Lien 57 : <http://labs.qt.nokia.com/2011/05/11/responses-to-qt-5/>
- Lien 58 : <https://launchpad.net/%7Eforumnokia/+archive/fin-ppa>
- Lien 59 : http://qt-project.org/wiki/Building_Qt_5_from_Git
- Lien 60 : <http://releases.qt-project.org/qt5.0/alpha/>
- Lien 61 : <http://qt-project.org/wiki/Ot-5-Alpha-building-instructions>
- Lien 62 : <http://www.developpez.net/forums/d1209764/c-cpp/bibliotheques/qt/sortie-qt-5-beta/>
- Lien 63 : <http://blog.qt.digia.com/2011/08/09/update-on-uikit-lighthouse-platform/>
- Lien 64 : <http://qt.developpez.com/actu/47705/Premier-jour-de-Qt-chez-Digia-apres-son-rachat-de-Nokia-le-framework-restera-disponible-sous-GPL-et-LGPL/>
- Lien 65 : <http://www.developpez.com/redirect/572>
- Lien 66 : <http://www.developpez.net/forums/d1248904/c-cpp/bibliotheques/qt/avenir-qt-developer-days/>
- Lien 67 : <http://alexandre-laurent.developpez.com/tutoriels/OpenGL/OpenGL-Extensions/#L2-B>
- Lien 68 : <http://freeglut.sourceforge.net/>
- Lien 69 : <http://qt-project.org/>
- Lien 70 : <http://qt-project.org/doc/qt-4.8/qtopenGL.html>
- Lien 71 : <http://qt-project.org/doc/qt-4.8/qglformat.html#setVersion>
- Lien 72 : <http://blog.developpez.com/gpu/p11241/langages-frameworks/opengl/qt-project.org/doc/qt-4.8/qglformat.html>
- Lien 73 : http://qt-project.org/wiki/How_to_use_OpenGL_Core_Profile_with_Qt
- Lien 74 : <http://www.sfml-dev.org/index-fr.php>
- Lien 75 : http://www.sfml-dev.org/documentation/2.0/structsf_1_1ContextSettings.php

Lien 76 : <http://www.sfml-dev.org/tutorials/2.0/window-opengl.php>
Lien 77 : <http://www.libsdl.org/>
Lien 78 : http://www.opengl.org/wiki/Tutorial1:_Creating_a_Cross_Platform_OpenGL_3.2_Context_in_SDL_%28C%2F_SDL%29
Lien 79 : <http://www.opengl.org/sdk/docs/man/xhtml/glsEnabled.xml>
Lien 80 : <http://www.opengl.org/sdk/docs/man/xhtml/glEnable.xml>
Lien 81 : <http://www.opengl.org/sdk/docs/man/xhtml/glGet.xml>
Lien 82 : http://blog.developpez.com/gpu/p11241/langages-frameworks/opengl/deboguer_avec_opengl
Lien 83 : <http://developer.nvidia.com/opengl-driver>
Lien 84 : <http://www.opengl.org/registry/doc/glspec43.core.20120806.pdf>
Lien 85 : <http://www.opengl.org/registry/doc/glspec43.compatibility.20120806.pdf>
Lien 86 : <http://www.opengl.org/registry/doc/GLSLangSpec.4.30.6.pdf>
Lien 87 : http://www.khronos.org/registry/gles/specs/3.0/es_spec_3.0.0.pdf
Lien 88 : http://www.khronos.org/registry/gles/specs/3.0/GLSL_ES_Specification_3.00.3.pdf
Lien 89 : <http://www.developpez.net/forums/d1250751/applications/developpement-2d-3d-jeux/api-graphiques/opengl/sortie-specifications-dopengl-4-3-opengl-es-3-0-a/>
Lien 90 : <http://jaxb.java.net/>
Lien 91 : <http://www.saxproject.org/>
Lien 92 : <http://www.oracle.com/technetwork/java/intro-140052.html>
Lien 93 : <http://docs.oracle.com/javase/6/docs/api/org/xml/sax/helpers/DefaultHandler.html>
Lien 94 : <http://www.castor.org/>
Lien 95 : <http://docs.oracle.com/javase/7/docs/api/java/beans/XMLEncoder.html>
Lien 96 : <http://atatorus.developpez.com/tutoriels/intro-jaxb/fichiers/XMLDecoder.html>
Lien 97 : <https://commons.apache.org/digester/>
Lien 98 : <http://jaxb.java.net/>
Lien 99 : <http://docs.oracle.com/javase/6/docs/api/java/util/LinkedHashMap.html>
Lien 100 : <http://jaxb.java.net/2.2.5/docs/>
Lien 101 : <http://atatorus.developpez.com/tutoriels/intro-jaxb/>
Lien 102 : <http://www.developpez.com/actu/47940/>
Lien 103 : <http://www.developpez.net/forums/d1266011/java/general-java/javaone-2012-demarre/>
Lien 104 : <http://www.developpez.com/actu/48206/>
Lien 105 : <http://www.developpez.net/forums/d1266773/java/general-java/javaone-2012-oracle-renouvelle-engagement-faire-evoluer-java-devoile-plans/>
Lien 106 : <http://www.developpez.com/actu/48228/>
Lien 107 : <http://www.jcp.org/en/jsr/detail?id=353>
Lien 108 : <http://www.developpez.net/forums/d1267436/java/general-java/javaone-2012-oracle-presente-specification-jsr-353-a/>
Lien 109 : <http://dlc.sun.com.edgesuite.net/netbeans/7.3/beta/>
Lien 110 : <http://www.developpez.net/forums/d1266726/java/edi-outils-java/netbeans/javaone-2012-oracle-sort-preview-netbeans-7-3-a/>
Lien 111 : <http://alain-bernard.developpez.com/tutoriels/eclipse/databinding-swt>
Lien 112 : <http://eclipse.developpez.com/cours/?page=platform-cat#plugin-dev>
Lien 113 : <http://eclipse.developpez.com/cours/?page=platform-cat#ihm-dev>
Lien 114 : <http://alain-bernard.developpez.com/tutoriels/eclipse/databinding-jface#L.II>
Lien 115 : <http://alain-bernard.developpez.com/tutoriels/eclipse/databinding-jface#L.III>
Lien 116 : <http://alain-bernard.developpez.com/tutoriels/eclipse/databinding-jface#L.IV>
Lien 117 : <http://www.vogella.de/articles/EclipseDataBinding/article.html>
Lien 118 : <http://alain-bernard.developpez.com/tutoriels/eclipse/databinding-jface/>
Lien 119 : <http://goo.gl/k9GD1>
Lien 120 : <http://goo.gl/DSDk7>
Lien 121 : <http://www.developpez.net/forums/d1267472/general-developpement/debats-developpement-best-of/android-os-geeks/>
Lien 122 : <http://www.developpez.net/forums/d1250546/club-professionnels-informatique/actualites/android-nouvelles-sources-menace-lhorizon-chercheurs-sappretent-publier-framework-dexpl/>