Developpez
Le Mag

Édition de Avril - Mai 2011.
Numéro 33.
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

Éditorial

Au printemps, les idées bourgeonnent !
Sur Developpez.com, cela produit encore plus d'articles qui satisferont votre soif de savoir.

Profitez-en bien !

La rédaction

Article XML

XPath 1.0 : fonctionnement des prédicats

Cette article étudie le fonctionnement des prédicats qui permettent de poser des conditions dans le langage XPath.

par **Erwan Amoureux**
Page 41

Article Perl

Concevoir facilement un plugin Nagios en Perl

Ce tutoriel a pour but de vous expliquer en quelques lignes comment concevoir un plugin Nagios respectant les normes Nagios avec Perl.

par **djibril**
Page 64

Java

Les dernières news

James Gosling (le père de Java) rejoint Google

L'annonce a été faite sur son blog ce matin. James Gosling est un nouvel employé de Google.

```
"I find myself starting employment at Google today. One of the toughest things about life is making choices. I had a hard time saying 'no' to a bunch of other excellent possibilities. I find it odd that this time I'm taking the road more travelled by, but it looks like interesting fun with huge leverage."
```

Cette annonce, qui prend par surprise la communauté Java, risque de faire couler beaucoup d'encre, puisque Google est toujours en litige avec Oracle.

Pour le moment, James Gosling ne donne (et n'a *a priori*) aucune information sur la nature de sa future tâche chez Google, même si beaucoup de facteurs laissent à penser que cette embauche a un rapport avec le désaccord récent entre Google et Oracle (Lien 01) sur l'utilisation de Java dans Android.

Pour mémoire, le père de Java surnomme Larry Ellison, PDG fondateur d'Oracle, le *"Prince des Ténèbres"* (parmi d'autres gentillesses sur Oracle : Lien 02). Il avait également été assez critique sur Android dont la fragmentation, à ses yeux, représente un frein au *"write once, use everywhere"* au cœur de Java.

James Gosling

Source : Lien 03

Commentez cette news de Selim KEBIR en ligne : Lien 04

Eclipse 3.7 sortira le 22 juin prochain

Et marquera un retour aux sources Java de la fondation Eclipse

Le traditionnel « Release Train », l'évènement durant lequel la fondation Eclipse présente chaque année (et le même jour) nombre de ses nouveautés technologiques aura lieu le 22 juin prochain.

Cette année, ce rendez-vous incontournable devrait marquer un retour aux sources Java de la fondation.

C'est en tout cas ce qu'a promis Mike Milinkovich, directeur exécutif de la fondation lors de la conférence EclipseCon 2011, organisée à Santa Clara, en Californie.

Eclipse 3.7, baptisé Indigo, devrait embarquer dès le mois de juin des fonctionnalités de la plateforme Java 7, comme les améliorations du langage introduites par le projet Coin, des nouveautés qui seront évidemment prises en charge par l'éditeur Java de l'IDE.

Indigo inclura la version 3.7 de la bibliothèque graphique « Standard Widget Toolkit » (SWT) et WindowBuilder, le créateur d'interfaces graphiques par Instantiations, offert par Google à la communauté Eclipse (Lien 05).

Indigo devrait aussi intégrer des outils Maven améliorés et la version 1.0 du projet « EGit », qui reliera l'espace de travail de l'IDE au système de contrôle de version distribué Git.

Indigo serait l'une des meilleures choses que la fondation ait faites pour les développeurs Java depuis des années, à en croire Mike Milinkovich.

Reste à savoir si les développeurs partageront son avis. Réponse le 22 juin prochain, donc.

Source : présentation de Mike Milinkovich (Lien 06)

Commentez cette news d'Idelways en ligne : Lien 07

Une application Android piratée lutte contre le piratage

Et humilie les utilisateurs qui l'installent

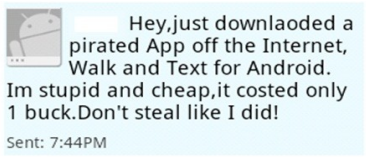
Une application Android se faisant passer pour une copie légitime de l'application « Walk and Text » circule sur les réseaux de partage de fichier P2P et... lutte contre le piratage des applications. L'application originale, disponible sur l'Android Market, permet d'écrire en marchant grâce à l'utilisation de la caméra du téléphone pour avoir une vision de la rue en arrière-plan de l'écran.

Se faisant passer pour une version non existante officiellement de l'application légitime (v 1.3.7), cette application installe le malware « Android.Walkinwat ».

Cependant, le but (inédit) de ce malware d'un autre genre n'est pas de prendre le contrôle de votre téléphone, ni même de compromettre les données personnelles et encore moins de voler des informations bancaires. Android.Walkinwat mène plutôt une guerre contre le piratage des applications.

En effet, après son exécution par un utilisateur, une boite de dialogue est présentée à celui-ci, lui donnant l'impression que l'application est compromise, alors que, en fait, en arrière-plan, l'application collecte les données sensibles de l'utilisateur et les envoie vers un serveur distant.

Ensuite, Android.Walkinwat va envoyer un SMS humiliant à tous les contacts de l'utilisateur comme « *hey, je viens de télécharger une application piratée depuis internet, Walk and Text pour Android. Je suis stupide et radin, elle coute à peine un dollar. Ne volez pas comme je le l'ai fait* ».



Et, pour finir, Android.Walkinwat affiche un dernier message à l'utilisateur lui demandant de vérifier sa facture et d'acheter les applications depuis la galerie officielle. « *Nous espérons vraiment que vous avez appris quelque chose de ceci. Vérifiez votre facture de téléphone. Oh et n'oubliez pas d'acheter l'application depuis le Market place* ».

Android.Walkinwat serait la première application du genre dans le paysage des menaces mobiles.

Source : Symantec (Lien 08)

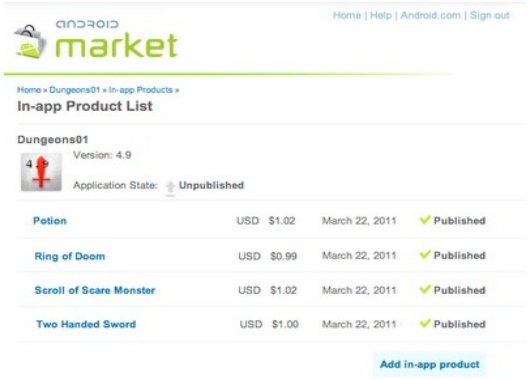
Commentez cette news d'Hinault Romaric en ligne : Lien 09

Android : le service d'achat à l'intérieur des applications officiellement lancé

Google prend une commission de 30% sur les ventes depuis In-App Billing

Comme prévu, In-App Billing, le nouveau service d'achat à l'intérieur des applications Android termine aujourd'hui sa phase de test et devient officiellement disponible pour les développeurs.

Concrètement, In-App Billing permet de procéder à une facturation des mises à jour, des fonctions additionnelles ou tout simplement des produits et contenus disponibles sans sortir de l'application.



Pour mémoire, Google a mis à jour la documentation relative à ce service, notamment sur la sécurité. Cette nouveauté oblige en effet les développeurs à redoubler d'attention sur ce point.

Tout comme pour la facturation d'application depuis l'Android Market, Google prend une commission de 30 % sur les ventes à l'intérieur des applications.

In-app Billing est supporté par les versions 1.6 et ultérieures d'Android.

Commentez cette news d'Hinault Romaric en ligne : Lien 10

Introduction à l'utilisation du Bluetooth dans une application Android

Cet article a pour objectif de vous apprendre à manipuler le périphérique Bluetooth, présent sur la plupart des terminaux Android, dans une application. Nous allons donc voir toutes les étapes de la création d'une liaison Bluetooth entre deux terminaux.

1. Introduction

Pour bien commencer, assurons-nous d'avoir le droit de l'utiliser et que le terminal qui exécute notre application possède un périphérique Bluetooth.

Demande de la permission d'utiliser le Bluetooth

```
<uses-permission
android:name="android.permission.BLUETOOTH" />
```

Vérification de la présence du Bluetooth sur le terminal

```
BluetoothAdapter blueAdapter =
BluetoothAdapter.getDefaultAdapter();
if (blueAdapter == null) {
    // Le terminal ne possède pas le Bluetooth
}
```

Avant d'être en mesure d'utiliser le Bluetooth dans votre application, il peut être de bonne pratique de vérifier si le Bluetooth est activé.

2. Activer le Bluetooth

Comme il est fort probable que le Bluetooth ne soit pas activé au moment voulu (beaucoup d'utilisateurs le désactivent pour éviter de recevoir des notifications de connexion ou tout simplement pour économiser leur batterie), voyons les deux manières possibles de faire pour l'allumer.

La première façon est de demander à l'utilisateur de le faire lui-même. Pour cela nous allons afficher une boîte de dialogue demandant à l'utilisateur s'il veut oui ou non activer son Bluetooth. Je vous rassure, pas la peine de coder vous-même la boîte de dialogue, le système sait le faire tout seul. Elle contiendra le message dans la langue du système de l'utilisateur ainsi que deux boutons, "Oui" et "Non".

Demande d'activation du Bluetooth

```
if (!blueAdapter.isEnabled()) {
    Intent enableBtIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBtIntent,
REQUEST_ENABLE_BT);
}
```

En utilisant "onActivityResult()", vous serez capable de savoir si l'utilisateur a cliqué sur oui ou non. Si le Bluetooth s'est correctement allumé, vous aurez un "RESULT_OK". Dans le cas contraire le retour sera "RESULT_CANCELED".

Vous pouvez également écouter l'Intent de broadcast "ACTION_STATE_CHANGED" pour savoir quand l'état du Bluetooth change. Vous pourrez alors utiliser "EXTRA_STATE" et "EXTRA_PREVIOUS_STATE" pour connaître l'état courant et l'état antérieur du Bluetooth. Les valeurs possibles sont "STATE_TURNING_ON", "STATE_ON", "STATE_TURNING_OFF" et "STATE_OFF".

La deuxième façon de faire est d'activer le Bluetooth directement, sans demander son consentement à l'utilisateur. Pour ce faire il va nous falloir demander une nouvelle permission.

Permission pour allumer et éteindre le Bluetooth

```
<uses-permission
android:name="android.permission.BLUETOOTH_ADMIN"
/>
```

Allumage du Bluetooth

```
if (!blueAdapter.isEnabled()) {
    blueAdapter.enable();
```

Activer la visibilité de votre terminal par Bluetooth active automatiquement le Bluetooth s'il n'est pas déjà actif (cf. plus loin).

3. Découvrir les périphériques

De même, activer la découverte des périphériques proches peut réduire de manière significative le débit d'une connexion Bluetooth préalablement établie.

Avant de chercher les périphériques à proximité pour une éventuelle connexion, commençons par regarder parmi les périphériques que nous connaissons. Pour cela nous allons appeler une fonction nommée "getBondedDevices()" qui renvoie une liste de "BluetoothDevice", chacun étant un périphérique connu.

Récupération des périphériques connus

```
Set<BluetoothDevice> pairedDevices =
blueAdapter.getBondedDevices();
```

Si le périphérique ne fait pas partie de ceux qui sont connus, il va falloir rechercher les terminaux visibles proches. Pour cela nous allons appeler "startDiscovery()", "cancelDiscovery()" permettant de stopper cette recherche. En général le processus dure une grosse dizaine de secondes, suivi d'une page qui affiche les périphériques

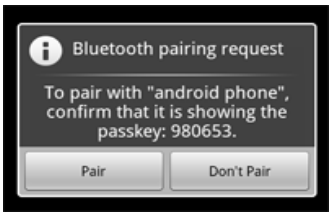
découverts. Voyons comment faire pour récupérer les résultats.

Découverte des terminaux Bluetooth visibles et proches

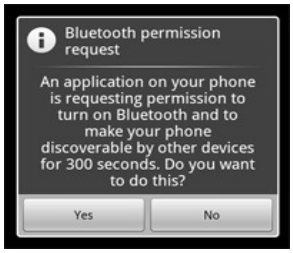
```
// On crée un BroadcastReceiver pour ACTION_FOUND
private final BroadcastReceiver receiver = new
BroadcastReceiver() {
  public void onReceive(Context context, Intent
intent) {
    String action = intent.getAction();
    // Quand la recherche trouve un terminal
    if
(BluetoothDevice.ACTION_FOUND.equals(action))
    {
      // On récupère l'object BluetoothDevice
depuis l'Intent
      BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_D
EVICE);
      // On ajoute le nom et l'adresse du
périphérique dans un ArrayAdapter (par exemple
pour l'afficher dans une ListView)
      mArrayAdapter.add(device.getName() + "\n" +
device.getAddress());
    }
  }
};
// Inscrire le BroadcastReceiver
IntentFilter filter = new
IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(receiver, filter); // N'oubliez
pas de le désinscrire lors du OnDestroy() !
```

La découverte des périphériques Bluetooth à proximité est très coûteuse en énergie. Pensez donc à appeler "cancelDiscovery" dès que possible.

Si les deux terminaux qui vont se connecter n'ont jamais été interconnectés, le système va automatiquement afficher une boîte de dialogue de confirmation. De ce fait vous n'avez pas à vous préoccuper au niveau de votre application de savoir si les terminaux se connaissent ou pas. La connexion va attendre que l'utilisateur valide la demande ou elle va échouer en cas de refus ou de timeout.



Par défaut, même si le Bluetooth est activé, les terminaux Android ne sont pas visibles. Pour activer la visibilité Bluetooth, nous allons procéder de la même manière que pour activer le Bluetooth. Nous allons demander à l'utilisateur de le faire par le biais d'une boîte de dialogue. Notez que par défaut la durée de visibilité est de 120 secondes, mais vous pouvez définir une durée de votre choix, inférieure ou égale à 300 secondes.



Demande à l'utilisateur l'autorisation pour rendre visible le terminal

```
Intent discoverableIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERAB
LE);
discoverableIntent.putExtra(BluetoothAdapter.EXTR
A_DISCOVERABLE_DURATION, timeVisible); // Cette
ligne permet de définir une durée de visibilité
de notre choix
startActivityForResult(discoverableIntent,
REQUEST_DISCOVERABLE_BT);
```

Une fois encore vous pourrez récupérer le choix de l'utilisateur dans le "onActivityResult()", de la même manière que précédemment. Un "RESULT_OK" vous assurera de la bonne visibilité de votre terminal.

Vous pouvez également être informé du changement de visibilité de votre terminal en inscrivant un "BroadcastReceiver" pour l'Intent "ACTION_SCAN_MODE_CHANGED". Les champs "EXTRA_SCAN_MODE" et "EXTRA_PREVIOUS_SCAN_MODE" vous donneront les informations sur l'état précédent et actuel de la visibilité du périphérique. Les valeurs possibles sont "SCAN_MODE_CONNECTABLE_DISCOVERABLE", "SCAN_MODE_CONNECTABLE", ou "SCAN_MODE_NONE".

Il n'est pas nécessaire pour un terminal d'être visible pour établir une connexion vers un autre terminal. Seul celui qui va héberger la connexion doit être visible pour que le client puisse le découvrir et initier la connexion.

4. Etablir une connexion côté serveur

La procédure à suivre est la suivante. Nous allons récupérer une "BluetoothServerSocket" en appelant "listenUsingRfcommWithServiceRecord(String, UUID)". Puis nous allons attendre une connexion par le biais de la méthode "accept()" de la socket. Une fois le(s) client(s) connecté(s), on bloque les demandes arrivant avec un appel à "close()".

Essayez de faire votre appel à la méthode "accept()" dans un Thread séparé car la méthode est bloquante. Cela aura pour effet de provoquer un ANR (Activity Not Responding) si le client met trop de temps à se connecter. Notez que toutes les méthodes des classes "BluetoothServerSocket" et "BluetoothSocket" sont "thread-safe".

Voici un thread simple qui accepte les connexions Bluetooth entrantes.

Exemple de thread Bluetooth serveur

```
private class ServeurBluetooth extends Thread {
  private final BluetoothServerSocket
blueServerSocket;

  public ServeurBluetooth() {
    // On utilise un objet temporaire qui sera
assigné plus tard à blueServerSocket car
blueServerSocket est "final"
    BluetoothServerSocket tmp = null;
    try {
      // MON_UUID est l'UUID (comprenez
identifiant serveur) de l'application. Cette
valeur est nécessaire côté client également !
      tmp =
blueAdapter.listenUsingRfcommWithServiceRecord(NO
M, MON_UUID);
    } catch (IOException e) { }
    blueServerSocket = tmp;
  }

  public void run() {
    BluetoothSocket blueSocket = null;
    // On attend une erreur ou une connexion
entrante
    while (true) {
      try {
        blueSocket = blueServerSocket.accept();
      } catch (IOException e) {
        break;
      }
      // Si une connexion est acceptée
      if (blueSocket != null) {
        // On fait ce qu'on veut de la connexion
(dans un thread séparé), à vous de la créer
        manageConnectedSocket(blueSocket);
        blueServerSocket.close();
        break;
      }
    }
  }

  // On stoppe l'écoute des connexions et on tue
le thread
  public void cancel() {
    try {
      blueServerSocket.close();
    } catch (IOException e) { }
  }
}
```

5. Etablir une connexion côté client

Voyons maintenant comment connecter un client à notre serveur. Nous allons supposer que nous avons déjà l'objet "BluetoothDevice" du terminal serveur (voir ci-avant).

Exemple de thread Bluetooth serveur

```
private class ClientBluetooth extends Thread {
  private final BluetoothSocket blueSocket;
  private final BluetoothDevice blueDevice;

  public ClientBluetooth(BluetoothDevice device)
{
    // On utilise un objet temporaire car
blueSocket et blueDevice sont "final"
    BluetoothSocket tmp = null;
    blueDevice = device;

    // On récupère un objet BluetoothSocket grâce
à l'objet BluetoothDevice
    try {
      // MON_UUID est l'UUID (comprenez
identifiant serveur) de l'application. Cette
valeur est nécessaire côté serveur également !
      tmp =
device.createRfcommSocketToServiceRecord(MON_UUID
);
    } catch (IOException e) { }
    blueSocket = tmp;
  }

  public void run() {
    // On annule la découverte des périphériques
(inutile puisqu'on est en train d'essayer de se
connecter)
    blueAdapter.cancelDiscovery();

    try {
      // On se connecte. Cet appel est bloquant
jusqu'à la réussite ou la levée d'une erreur
      blueSocket.connect();
    } catch (IOException connectException) {
      // Impossible de se connecter, on ferme la
socket et on tue le thread
      try {
        blueSocket.close();
      } catch (IOException closeException) { }
      return;
    }

    // Utilisez la connexion (dans un thread
séparé) pour faire ce que vous voulez
    manageConnectedSocket(blueSocket);
  }

  // Annule toute connexion en cours et tue le
thread
  public void cancel() {
    try {
      blueSocket.close();
    } catch (IOException e) { }
  }
}
```

Retrouvez l'article de Sylvain Berfini en ligne : Lien 11

Utilisation du NFC sous Android pour établir une connexion Bluetooth

Cet article a pour but d'utiliser le capteur NFC présent sur certains AndroPhones afin de permettre à deux terminaux équipés d'établir une connexion Bluetooth sans faire intervenir l'utilisateur.

1. Prérequis

Afin que vous ne soyez pas perdu dans la lecture de ce tutoriel, il est indispensable qu'en plus des bases dans la programmation d'applications Android vous sachiez utiliser le capteur NFC pour envoyer et recevoir des données. Si ce n'est pas le cas je vous invite à lire attentivement cet article : Introduction à l'utilisation du NFC dans une application Android (Lien 12).

La connaissance dans l'utilisation du Bluetooth en programmation Android n'est pas nécessaire en elle-même, cependant elle serait souhaitable. Vous pouvez si vous le souhaitez commencer par ce tutoriel : Introduction à l'utilisation du Bluetooth dans une application Android (Lien 11).

2. Préambule

Depuis la release 2.3.3 (API 10) du SDK d'Android, certaines nouvelles méthodes très pratiques ont fait leur apparition. Outre les améliorations de l'API pour ce qui concerne le NFC, il ne faut pas oublier la possibilité de créer/se connecter à un serveur (socket) Bluetooth non sécurisé.

Quel est l'avantage d'avoir un serveur non sécurisé alors qu'avant nous en avions un sécurisé ? Eh bien nous n'avons plus besoin de faire participer l'utilisateur pour se connecter à un autre terminal en Bluetooth. Il suffit de connaître l'UUID de la socket, l'adresse MAC du périphérique Bluetooth du terminal et bien sûr d'être à portée. Et puisque ces deux informations s'avèrent n'être ni plus ni moins que des chaînes de caractères, il est très facile et peu coûteux (en taille) de les faire naviguer.

Et c'est là que le NFC entre en jeu. Puisque de toute façon il faut être proche (au sens "humain") pour instaurer un dialogue en Bluetooth, autant se servir du NFC dont l'inconvénient majeur (la portée) n'en est plus un. L'objectif de ce tutoriel va donc être de configurer et synchroniser une socket Bluetooth simplement en rapprochant deux terminaux équipés d'un capteur NFC.

3. Squelette de l'application

Après ce préambule, entrons maintenant dans les détails techniques. Je vais créer trois classes héritant d'Activity au sein de mon projet.

L'une (que j'appellerai "hôte") devra être lancée par un des deux terminaux qui créera une socket Bluetooth de type serveur et émettra un Tag (par Ndef Push) contenant l'UUID du serveur ainsi que son adresse MAC Bluetooth. Elle pourra être lancée depuis le menu par l'utilisateur voulant être l'hôte de la session.

La deuxième (dite "cliente") attendra la découverte d'un Tag pour se lancer, et tentera de se connecter au serveur dont il récupérera les informations dans le Tag reçu. Elle pourra également être lancée par l'utilisateur depuis le menu et lui permettra de tenter de se connecter à un couple adresse MAC/UUID connu lors d'une précédente session.

La dernière sera l'Activity principale, dite "menu", c'est-à-dire que lors d'une pression sur l'icône de mon application, ce sera elle qui sera lancée.

Mais avant de rentrer dans le code de celles-ci, voyons la partie "importante" du fichier Manifest.xml. Nous allons y déclarer nos trois Activity : le menu comme action Main/Launcher, le client comme action Tag_Discovered et l'hôte comme une Activity toute simple. Pour terminer, nous allons y déclarer les permissions dont nous allons avoir besoin, à savoir celle pour gérer le capteur NFC (sans oublier le filtre pour interdire l'installation de notre application sur les terminaux ne possédant pas de capteur NFC), ainsi que celles pour administrer (comprendre allumer/éteindre) et utiliser le Bluetooth.

Manifest.xml

```
<application android:icon="@drawable/icon"
android:label="@string/app_name">
  <activity android:name=".Menu"
android:label="@string/app_name">
    <intent-filter>
      <action
android:name="android.intent.action.MAIN" />
      <category
android:name="android.intent.category.LAUNCHER" /
>
    </intent-filter>
  </activity>
  <activity android:name=".Host"
android:label="Host"/>
  <activity android:name=".Client"
android:label="Client">
    <intent-filter>
      <action
android:name="android.nfc.action.TAG_DISCOVERED"/
>
      <category
android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
  </activity>
</application>
<uses-sdk android:minSdkVersion="10" />
<uses-feature android:name="android.hardware.nfc"
android:required="true" />
<uses-permission
android:name="android.permission.NFC" />
<uses-permission
android:name="android.permission.BLUETOOTH" />
<uses-permission
android:name="android.permission.BLUETOOTH_ADMIN"
/>
```

Maintenant voyons le code de nos Activity. Je ne vais pas vous coller tout le contenu de chacune des trois classes (parce que cela ne vous apporterait rien sur le plan de l'apprentissage), mais juste les parties du code qui

concernent l'interaction NFC/Bluetooth.

4. Activity hôte

Si vous avez lu mes précédents tutoriels sur le NFC et sur le Bluetooth, rien ici ne devrait vous choquer. Je récupère l'adresse MAC de mon périphérique Bluetooth après avoir fait en sorte qu'il soit bien activé, puis je crée un NdefMessage qui contient cette adresse ainsi que l'UUID de mon serveur Bluetooth et je le push en utilisant le capteur NFC. Ensuite je crée la socket serveur Bluetooth et j'attends une connexion. La suite ne dépend plus que de vous...

Activity Hote

```
public class Hote extends Activity
{
  private static final String BT_SERVER_UUID_INSECURE = "8ce255c0-200a-11e0-ac64-0800200c9a66"; // A vous d'en choisir un
  private NfcAdapter nfcAdapter;
  private BluetoothAdapter blueAdapter;
  private NdefRecord blueMessage;
  private BluetoothSocket blueSocket;

  public void onCreate(Bundle savedInstanceState)
  {
  ...
  }

  private NdefRecord createRecord(String msg)
  {
    byte[] langBytes = Locale.ENGLISH.getLanguage().getBytes(Charset.forName("US-ASCII"));
    byte[] textBytes = msg.getBytes(Charset.forName("UTF-8"));
    char status = (char) (langBytes.length);

    byte[] data = new byte[1 + langBytes.length + textBytes.length];
    data[0] = (byte) status;
    System.arraycopy(langBytes, 0, data, 1, langBytes.length);
    System.arraycopy(textBytes, 0, data, 1 + langBytes.length, textBytes.length);

    return new NdefRecord(NdefRecord.TNF_WELL_KNOWN, NdefRecord.RTD_TEXT, new byte[0], data);
  }

  protected void onPause()
  {
    super.onPause();
    if(NfcAdapter.getDefaultAdapter(this) != null)
    {
      NfcAdapter.getDefaultAdapter(this).disableForegroundNdefPush(this);
    }
  }

  protected void onResume()
  {
    super.onResume();

    nfcAdapter = NfcAdapter.getDefaultAdapter(this);
    blueAdapter = BluetoothAdapter.getDefaultAdapter();
    if (nfcAdapter == null || blueAdapter == null)
    {
      finish();
      return;
    }
    if (!blueAdapter.isEnabled())
      blueAdapter.enable();

    try
    {
      UUID blueUUID = UUID.fromString(BT_SERVER_UUID_INSECURE);
      BluetoothServerSocket blueServerSocket = blueAdapter.listenUsingInsecureRfcommWithServiceRecord("MyBluetoothChatService", blueUUID);
      String blueAddress = blueAdapter.getAddress();

      blueMessage = createRecord(blueAddress + "$" + BT_SERVER_UUID_INSECURE);
      NdefRecord[] rec = new NdefRecord[1];
      rec[0] = record;
      NdefMessage msg = new NdefMessage(rec);
      nfcAdapter.enableForegroundNdefPush(this, msg);

      // Je mets cet appel ici mais il serait préférable de le faire dans un Thread différent (l'appel est bloquant)
      blueSocket = blueServerSocket.accept();
      ...
    }
    catch (Exception e)
    {
      e.printStackTrace();
    }
  }
}
```

5. Activity cliente

De la même façon, j'ai ici créé une Activity s'exécutant lors de la découverte d'un Tag et qui vérifie que le Bluetooth est bien activé (l'activant si nécessaire). Ensuite elle analyse le message reçu, en déduit l'adresse MAC et l'UUID du serveur Bluetooth et tente de s'y connecter.

Activity cliente

```
public class Client extends Activity
{
  private NfcAdapter nfcAdapter;
  private BluetoothAdapter blueAdapter;
  private BluetoothSocket blueSocket;

  public void onCreate(Bundle savedInstanceState)
  {
    ...

    nfcAdapter = NfcAdapter.getDefaultAdapter(this);
    if (nfcAdapter == null)
    {
      Toast.makeText(this, "Votre terminal n'est pas équipé d'un capteur NFC !", Toast.LENGTH_SHORT).show();
      System.exit(0);
    }
```

```
    resolveIntent(getIntent());
  }

  private String getBluetoothAddress(String
msg) // Renvoie l'adresse MAC du serveur
  {
    return msg.substring(0, 17);
  }

  private UUID getUUID(String msg) throws
NumberFormatException // Renvoie l'UUID du
serveur
  {
    return UUID.fromString(msg.substring(18)); //
On commence à 18 pour sauter l'adresse MAC et le
caractère de délimitation que j'ai utilisé dans
la classe Hote
  }

  private String getText(final byte[] payload)
  {
    if (payload == null) return null;
    try
    {
      String textEncoding = ((payload[0] & 0200)
== 0) ? "UTF-8" : "UTF-16";
      int languageCodeLength = payload[0] & 0077;
      return new String(payload,
languageCodeLength + 1, payload.length -
languageCodeLength - 1, textEncoding);
    }
    catch (Exception e)
    {
      e.printStackTrace();
    }
    return null;
  }

  void resolveIntent(Intent intent)
  {
    String action = intent.getAction();
    if
(NfcAdapter.ACTION_TAG_DISCOVERED.equals(action))
    {
      Parcelable[] rawMsgs =
intent.getParcelableArrayExtra(NfcAdapter.EXTRA_N
DEF_MESSAGES);
      NdefMessage[] msgs;
      if (rawMsgs != null)
      {
        msgs = new NdefMessage[rawMsgs.length];
        for (int i = 0; i < rawMsgs.length; i++)
        {
          msgs[i] = (NdefMessage) rawMsgs[i];
        }

        try
        {
          NdefRecord record =
msgs[0].getRecords()[0];
          String message =
getText(record.getPayload());
          String blueAddress =
getBluetoothAddress(message);
          UUID blueUUID = getUUID(message);

          blueAdapter =
BluetoothAdapter.getDefaultAdapter();
          while (!blueAdapter.isEnabled())
            blueAdapter.enable();
          BluetoothDevice blueDevice =
blueAdapter.getRemoteDevice(blueAddress);
          blueSocket =
blueDevice.CreateInsecureRfcommSocketToServiceRec
ord(blueUUID);
          blueSocket.connect();

          // Vous êtes connecté, à vous de faire
ce que vous voulez
          ...
        }
        catch (Exception e)
        {
          e.printStackTrace();
        }
      }
    }
  }
}
```

Retrouvez l'article de Sylvain Berfini en ligne : Lien 13

Comment utiliser SQLite sous Android

Cet article a pour objectif de vous apprendre à utiliser SQLite sous Android. C'est toujours utile de pouvoir stocker quelques données dans une application. Les tables que l'on crée avec SQLite doivent bien entendu être simples, tout comme les requêtes que vous ferez dessus. Rappelez-vous que l'on n'est pas sur des mégaserveurs ultrapuissants, mais simplement sur un mobile.

1. Introduction

Pour notre exemple, nous allons créer une mini base de données pour enregistrer des livres. Durant ce tutoriel, nous utiliserons les classes suivantes :

- SQLiteOpenHelper qui est une classe d'assistance pour gérer la création de bases de données et la gestion des versions (Lien 14) ;
- ContentValue (Lien 15), cette classe est utilisée pour stocker un ensemble de valeurs que le ContentResolver (Lien 16) peut traiter ;
- Cursor est une interface qui donne accès en lecture-écriture à l'ensemble des résultats retournés par une requête de base de données (Lien 17).

On va donc commencer doucement avec la création de la classe Livre (très simple pour notre exemple). D'ailleurs, dernière petite précision, dans ce tutoriel on ne va pas faire d'interface graphique, on va juste afficher les résultats de nos requêtes dans des toasts.

2. Code Java

Après avoir créé un nouveau projet (perso moi je suis toujours avec Android 1.6), on va tout de suite concevoir une nouvelle classe que l'on va appeler **Livre**. Cette classe est très simple puisque dans notre cas, un livre est défini

par un ID, un numéro ISBN et un titre. On crée le constructeur ainsi que les getter et les setter et le tour est joué. Voici le code :

```
package com.tutomobile.android.sqlite;

/**
 * Création d'un livre tout simple pour un exemple d'utilisation de SQLite sous Android
 * @author Axon
 * http://www.tutomobile.fr
 */
public class Livre {

  private int id;
  private String isbn;
  private String titre;

  public Livre(){}

  public Livre(String isbn, String titre){
    this.isbn = isbn;
    this.titre = titre;
  }

  public int getId() {
    return id;
  }

  public void setId(int id) {
    this.id = id;
  }

  public String getIsbn() {
    return isbn;
  }

  public void setIsbn(String isbn) {
    this.isbn = isbn;
  }

  public String getTitre() {
    return titre;
  }

  public void setTitre(String titre) {
    this.titre = titre;
  }

  public String toString(){
    return "ID : "+id+"\nISBN : "+isbn+"\nTitre : "+titre;
  }
}
```

Maintenant nous allons faire une nouvelle classe que j'ai appelée **MaBaseSQLite** et qui hérite de **SQLiteOpenHelper**. Cette classe va nous permettre de définir la table qui sera produite lors de l'instanciation de celle-ci. Le code est très simple vous allez voir, et je l'ai commenté :

```
package com.tutomobile.android.sqlite;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteDatabase.CursorFactory;

public class MaBaseSQLite extends SQLiteOpenHelper {

  private static final String TABLE_LIVRES = "table_livres";
  private static final String COL_ID = "ID";
  private static final String COL_ISBN = "ISBN";
  private static final String COL_TITRE = "Titre";

  private static final String CREATE_BDD = "CREATE TABLE " + TABLE_LIVRES + " ("
  + COL_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " + COL_ISBN + " TEXT NOT NULL, "
  + COL_TITRE + " TEXT NOT NULL);";

  public MaBaseSQLite(Context context, String name, CursorFactory factory, int version) {
    super(context, name, factory, version);
  }

  @Override
  public void onCreate(SQLiteDatabase db) {
    //on crée la table à partir de la requête écrite dans la variable CREATE_BDD
    db.execSQL(CREATE_BDD);
  }

  @Override
  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    //On peut faire ce qu'on veut ici moi j'ai décidé de supprimer la table et de la recréer
    //comme ça lorsque je change la version les id repartent de 0
   db.execSQL("DROP TABLE " + TABLE_LIVRES + ";");
    onCreate(db);
  }
}
```

Ensuite nous allons créer une nouvelle classe (rassurez-vous, c'est la dernière) que j'ai appelée **LivresBDD**. Elle va nous permettre de gérer l'insertion, la suppression, la modification de livres dans la BDD (Base De Données) ainsi que de faire des requêtes pour récupérer un livre contenu dans la base de données. Comme d'habitude je vous donne le code commenté, j'espère que cela suffira pour comprendre :

```
package com.tutomobile.android.sqlite;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;

public class LivresBDD {

  private static final int VERSION_BDD = 1;
  private static final String NOM_BDD = "eleves.db";

  private static final String TABLE_LIVRES = "table_livres";
  private static final String COL_ID = "ID";
  private static final int NUM_COL_ID = 0;
```

```
  private static final String COL_ISBN = "ISBN";
  private static final int NUM_COL_ISBN = 1;
  private static final String COL_TITRE =
"Titre";
  private static final int NUM_COL_TITRE = 2;

  private SQLiteDatabase bdd;

  private MaBaseSQLite maBaseSQLite;

  public LivresBDD(Context context){
    //On crée la BDD et sa table
    maBaseSQLite = new MaBaseSQLite(context,
NOM_BDD, null, VERSION_BDD);
  }

  public void open(){
    //on ouvre la BDD en écriture
    bdd = maBaseSQLite.getWritableDatabase();
  }

  public void close(){
    //on ferme l'accès à la BDD
    bdd.close();
  }

  public SQLiteDatabase getBDD(){
    return bdd;
  }

  public long insertLivre(Livre livre){
    //Création d'un ContentValues (fonctionne
comme une HashMap)
    ContentValues values = new ContentValues();
    //on lui ajoute une valeur associée à une clé
(qui est le nom de la colonne dans laquelle on
veut mettre la valeur)
    values.put(COL_ISBN, livre.getIsbn());
    values.put(COL_TITRE, livre.getTitre());
    //on insère l'objet dans la BDD via le
ContentValues
    return bdd.insert(TABLE_LIVRES, null,
values);
  }

  public int updateLivre(int id, Livre livre){
    //La mise à jour d'un livre dans la BDD
fonctionne plus ou moins comme une insertion
    //il faut simplement préciser quel livre on
doit mettre à jour grâce à l'ID
    ContentValues values = new ContentValues();
    values.put(COL_ISBN, livre.getIsbn());
    values.put(COL_TITRE, livre.getTitre());
    return bdd.update(TABLE_LIVRES, values,
COL_ID + " = " +id, null);
  }

  public int removeLivreWithID(int id){
    //Suppression d'un livre de la BDD grâce à
l'ID
    return bdd.delete(TABLE_LIVRES, COL_ID + " =
" +id, null);
  }

  public Livre getLivreWithTitre(String titre){
    //Récupère dans un Cursor les valeurs
correspondant à un livre contenu dans la BDD (ici
on sélectionne le livre grâce à son titre)
    Cursor c = bdd.query(TABLE_LIVRES, new
String[] {COL_ID, COL_ISBN, COL_TITRE}, COL_TITRE
+ " LIKE \"" + titre +"\"", null, null, null,
null);
    return cursorToLivre(c);
  }

  //Cette méthode permet de convertir un cursor
en un livre
  private Livre cursorToLivre(Cursor c){
    //si aucun élément n'a été retourné dans la
requête, on renvoie null
    if (c.getCount() == 0)
      return null;

    //Sinon on se place sur le premier élément
    c.moveToFirst();
    //On créé un livre
    Livre livre = new Livre();
    //on lui affecte toutes les infos grâce aux
infos contenues dans le Cursor
    livre.setId(c.getInt(NUM_COL_ID));
    livre.setIsbn(c.getString(NUM_COL_ISBN));
    livre.setTitre(c.getString(NUM_COL_TITRE));
    //On ferme le cursor
    c.close();

    //On retourne le livre
    return livre;
  }
}
```

Bon allez encore un petit d'effort, on tient le bon bout, il ne reste plus qu'à faire le petit bout de code de test. Je ne suis même pas obligé de vous le donner, car si vous avez bien compris ce qu'on a fait avant vous pourrez le faire les doigts dans le nez. Mais bon si vous n'avez pas compris ce qu'on a fait avant, c'est peut-être que j'ai mal expliqué donc je vais vous donner le code pour tester notre programme. Celui-ci est à mettre dans l'Activity qui se crée par défaut lorsque vous faites votre nouveau projet Android :

```
package com.tutomobile.android.sqlite;

import android.app.Activity;
import android.os.Bundle;
import android.widget.Toast;

public class Tutorial16_Android extends Activity
{
  /** Called when the activity is first created.
*/
  @Override
  public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    //Création d'une instance de ma classe
LivresBDD
    LivresBDD livreBdd = new LivresBDD(this);

    //Création d'un livre
    Livre livre = new Livre("123456789",
"Programmez pour Android");

    //On ouvre la base de données pour écrire
dedans
    livreBdd.open();
    //On insère le livre que l'on vient de créer
    livreBdd.insertLivre(livre);
```

```
    //Pour vérifier que l'on a bien créé notre livre dans la BDD
    //on extrait le livre de la BDD grâce au titre du livre que l'on a créé précédemment
    Livre livreFromBdd = livreBdd.getLivreWithTitre(livre.getTitre());
    //Si un livre est retourné (donc si le livre à bien été ajouté à la BDD)
    if(livreFromBdd != null){
      //On affiche les infos du livre dans un Toast
      Toast.makeText(this, livreFromBdd.toString(), Toast.LENGTH_LONG).show();
      //On modifie le titre du livre
      livreFromBdd.setTitre("J'ai modifié le titre du livre");
      //Puis on met à jour la BDD
      livreBdd.updateLivre(livreFromBdd.getId(), livreFromBdd);
    }

    //On extrait le livre de la BDD grâce au nouveau titre
    livreFromBdd = livreBdd.getLivreWithTitre("J'ai modifié le titre du livre");
    //S'il existe un livre possédant ce titre dans la BDD
    if(livreFromBdd != null){
      //On affiche les nouvelles informations du livre pour vérifier que le titre du livre a bien été mis à jour
      Toast.makeText(this, livreFromBdd.toString(), Toast.LENGTH_LONG).show();
      //on supprime le livre de la BDD grâce à son ID
      livreBdd.removeLivreWithID(livreFromBdd.getId());
    }

    //On essaye d'extraire de nouveau le livre de la BDD toujours grâce à son nouveau titre
    livreFromBdd = livreBdd.getLivreWithTitre("J'ai modifié le titre du livre");
    //Si aucun livre n'est retourné
    if(livreFromBdd == null){
      //On affiche un message indiquant que le livre n'existe pas dans la BDD
      Toast.makeText(this, "Ce livre n'existe pas dans la BDD", Toast.LENGTH_LONG).show();
    }
    //Si le livre existe (mais normalement il ne devrait pas)
    else{
      //on affiche un message indiquant que le livre existe dans la BDD
      Toast.makeText(this, "Ce livre existe dans la BDD", Toast.LENGTH_LONG).show();
    }
    livreBdd.close();
  }
}
```

Si vous lancez votre application, vous devriez voir apparaître successivement les écrans suivants :

Android SQLite

Android SQLite

Android SQLite

Voilà ! On en a enfin fini avec ce tutoriel sur SQLite ! Alors c'est si terrible que ça ? Si vous n'avez pas compris quelque chose, n'hésitez pas à poser vos questions dans un commentaire (enfin regardez bien d'abord si vous avez tout fait ;)).

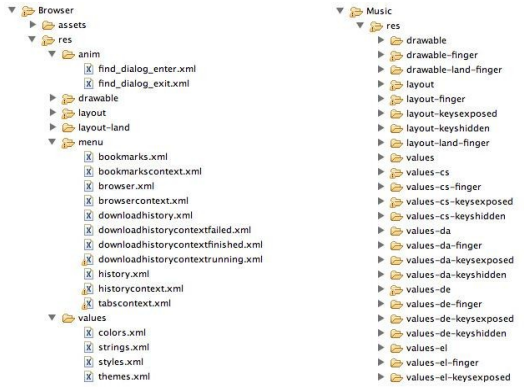
Retrouvez l'article d'Axon de Tuto Mobile en ligne : Lien 18

Extrait du livre "Développez pour Android" la spécialisation des ressources

Cet article est un extrait du livre "Développez pour Android". Il présente les ressources mises en œuvre dans le cadre du développement d'une application Android : la spécialisation des ressources, les ressources Android, les ressources non structurées et la création d'une bibliothèque de ressources.

1. Spécialisation des ressources

Les exemples de spécialisation des ressources les plus complets se trouvent dans le code source d'Android ; vous trouverez ci-dessous la présentation d'une partie des ressources du navigateur web (à gauche) et du lecteur de musique (à droite).



Arborescence des ressources des applications standard navigateur et lecteur de musique

Pour le navigateur web, remarquez les ressources par défaut values ou layout . layout-land s'appliquera uniquement lorsque le périphérique est en mode paysage. Pour le lecteur de musique, notez les différents dossiers values caractérisés par la langue (cs, da, etc.), mais également par la présence d'un clavier coulissant en position ouverte ou fermée ou bien d'un écran tactile utilisable avec le doigt (par opposition à un écran à stylet). Spécialiser des ressources implique donc la création de sous-dossiers spécifiques, dont le nom doit correspondre au protocole spécifié par Google et dont une liste exhaustive est donnée ci-après. Ce mode opératoire peut se voir appliqué pour l'ensemble des dossiers présents dans res . Les méthodes d'accès décrites précédemment s'appliqueront à l'ensemble des ressources et, pour un même identifiant, c'est le système qui déterminera quelle ressource choisir en fonction des critères. Ci-après, nous listons les principaux critères susceptibles d'être pris en compte quant à la spécification des ressources à utiliser lors de l'exécution de l'application. Une application s'affichera sans erreur à condition que les ressources soient au minimum écrites pour les valeurs par défaut dans les dossiers, comme layout, values, menu, etc. Ensuite, il est possible d'utiliser les critères pour améliorer l'expérience utilisateur (langages, interfaces).

1.1. Les langages et régions

Comme indiqué dans l'exemple String Format (figure 5-1), un langage est décrit par deux lettres, et ce conformément à la norme ISO 639-1. À titre d'exemple, fr correspond au français, en à l'anglais et de à l'allemand. Certaines variations linguistiques pouvant être observées pour une même langue selon la région où celle-ci est pratiquée, un critère régional peut venir compléter le type de langage. Ainsi, deux lettres correspondant à la région considérée et répondant à la norme ISO 3166-1-alpha-2, précédées de la lettre r peuvent être ajoutées de manière à affiner la spécification, par exemple dans l'exemple 5-3 nous avons différencié l'anglais pratiqué aux États-Unis et en Angleterre avec le suffixe en-rGB.

Exemple : un dossier positionné au même niveau que values et nommé values-fr contient les valeurs à utiliser dans le cas où la langue choisie par l'utilisateur pour son périphérique est le français. Dans le cas où aucun dossier ne correspond à la langue courante, les valeurs contenues dans values sont prises en considération par défaut. Ce critère de langage s'applique également aux couleurs, styles et thèmes présents dans le dossier values .

1.2. Les codes réseaux nationaux et commerciaux

Les ressources peuvent être différenciées selon le critère du réseau sur lequel l'application est exécutée. Cette caractéristique se décompose en un premier code correspondant au réseau du pays, défini par les lettres mcc suivies du code pays, et un deuxième code correspondant à l'opérateur sollicité, défini par les lettres mnc , suivies du code opérateur (on trouve facilement le code des opérateurs mondiaux sur Internet).

Exemple : imaginons une application qui se comporte différemment selon les opérateurs, par exemple pour l'affichage d'un logo ou d'une grille tarifaire des services proposés par celui-ci.

1.3. Les tailles des écrans

L'un des critères essentiels quant au choix des ressources à mettre en oeuvre, lors de l'exécution de l'applicatif Android, réside dans la taille de l'écran qui équipe le terminal. Quatre tailles sont identifiables : petit, normal et grand et très grand : small , normal , large , xlarge (introduit avec la version 2.3).

Exemple : le critère de taille d'écran permet de définir des layouts différents. Il est possible d'afficher plus ou moins d'informations sur un layout ou même d'avoir une disposition et une ergonomie différente.

1.4. Les formats d'écrans

La longueur de l'écran, en termes d'aspect ratio, peut également être considérée. Il s'agit là de la caractéristique Screen Aspect, pouvant avoir pour valeur long ou notlong .

Exemple : comme pour le critère précédent, l'expérience utilisateur peut être différenciée suivant le format de l'écran.

1.5. La densité des pixels des écrans

La densité pixellique de l'écran peut également être prise en considération. Pour ce faire, plusieurs niveaux de résolution sont identifiables : ldpi (basse résolution), mdpi (résolution moyenne), hdpi (haute résolution), xhdpi (très haute résolution) ou encore nodpi (utilisé pour empêcher la mise à l'échelle des images).

Exemple : permet d'optimiser l'utilisation des images en fonction des capacités de l'écran, ainsi il est inutile d'afficher une image haute résolution sur un écran qui ne la supporte pas. L'exemple le plus courant concerne les icônes des applications.

Les trois formats d'icône utilisés pour les écrans basse, moyenne et haute résolution ;ces images sont tirées du projet DigitbooksAverageRating

1.6. L'orientation de l'écran

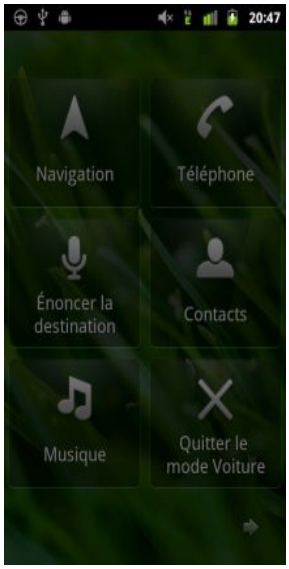
Il est envisageable, voire conseillé, de définir des ressources spécifiques en fonction de l'orientation de l'écran lors de l'utilisation de l'applicatif développé. Pour ce faire, le développeur peut utiliser la caractéristique screen orientation avec pour possibilités land (mode paysage) et port (mode portrait).

Exemple : certains périphériques se prêtent à l'utilisation portrait ou paysage, il peut donc être intéressant de différencier le comportement des layouts pour améliorer l'expérience utilisateur.

1.7. Les contextes d'utilisation des terminaux

Un terminal Android peut définir son contexte d'utilisation pour choisir des ressources particulières en fonction de la manière dont il est utilisé. Le terminal utilise des capteurs pour déterminer le contexte actuel. Ces contextes sont au nombre de trois : le premier étant le mode normal , le second est le mode voiture, qui s'active lors de la pose du terminal sur un support de type voiture, et le dernier, le mode bureau lorsque l'on utilise un support de bureau (car et desk).

Exemple : l'application Home d'Android présente des modes différents avec, par exemple, dans le cas du support voiture un accès simplifié à la navigation, la musique, le bluetooth ou la reconnaissance vocale.



Dans le cas du mode bureau, les informations sur l'heure, les éphémérides ou encore la météo sont affichées en permanence.

Application Home en mode bureau sur un Motorola Milestone (il possède un support de bureau physique qui active ce mode)

1.8. Les critères de situation temporelle

Les modes diurne et nocturne peuvent également être spécifiés (night / notnight). Ils s'activeront manuellement ou automatiquement dans le cas où l'utilisateur est en mode voiture en fonction de l'heure du téléphone.

Exemple : imaginons un changement des couleurs de l'interface de l'écran pour éviter l'éblouissement du conducteur.

1.9. Les types d'écrans tactiles

Deux types d'écrans tactiles sont identifiables au niveau de la spécification des ressources. Il s'agit des écrans résistifs à stylets, identifiés par le terme stylus , et des écrans, généralement capacitifs, utilisables au travers des doigts, identifiés par le terme finger . Les ressources à mettre en oeuvre dans le cas de terminaux dépourvus d'écran tactile peuvent être répertoriées dans le dossier notouch .

Exemple : d'une importance capitale pour l'expérience utilisateur, il faut par exemple prendre en compte la taille des doigts pour les interfaces qui se pilotent au doigt, alors que les stylets permettront des zones tactiles plus petites et laisseront plus de place pour les contenus.

1.10. Présence d'un clavier physique et son état

Nous pouvons d'une part, caractériser la présence d'un clavier physique keyssoft sur le périphérique, mais également son état qui peut évoluer au cours de la vie de l'application keysexposed et keyshidden .

Exemple : lorsque le clavier physique est caché, il faut penser que le clavier virtuel vient recouvrir une bonne partie de l'interface utilisateur.

1.11. Type de clavier physique

Aucun avec nokeys , qwerty pour les claviers complets, 12key pour les claviers numériques.

1.12. Type des touches de navigation

Lorsque le téléphone n'a pas de touche de navigation, le critère est nonav . Dans le cas contraire, le périphérique de navigation peut prendre aujourd'hui plusieurs formes. Pad directionnel à quatre touches avec dpad , roulette avec wheel pour un défilement, boule roulante équivalente au dpad , mais moins encombrant sur le périphérique avec trackball .

Exemple : essentiel pour les jeux, le périphérique de navigation permet la plupart du temps de diriger le personnage principal du jeu. Donc s'il n'y en a pas, il doit être remplacé par un joystick virtuel (affiché à l'écran par exemple). S'il est présent, l'interprétation des actions utilisateur doit être différente pour un pad ou un trackball si un personnage se déplace à l'écran.

1.13. État des touches de navigation

Comme pour le clavier, le système de navigation peut être caché derrière un clapet ou autre, nous avons donc les critères navexposed et navhidden.

1.14. La version du système

Il est envisageable de prendre en considération le système en termes de version API, au travers de dossiers spécifiques (comme v4 à partir d'Android 1.6).

Exemple : permet d'éviter l'utilisation de widgets dépréciés ou même d'améliorer l'interface avec de nouveaux widgets fournis par les nouvelles versions du SDK.

2. Ressources Android

Dans le but de créer une homogénéité sur certaines parties des applications, la bibliothèque jar Android embarque un certain nombre de ressources qui peuvent être utilisées dans les applications. Nous accédons aux identifiants de ces ressources par la classe android.R. , suivie des types que nous avons vus ci-dessus.

Dans ces ressources, sont définis des comportements standard, comme les animations de fondu entrant ou sortant (android.R.anim.fade_in et android.R.anim.fade_out), des constantes, comme la durée des animations, les dimensions des vignettes et des icônes du système.

On trouvera également des images, comme les icônes de menu pour les actions standard : ajouter, supprimer, partager, recherche.

Quelques icônes présentes dans les ressources du SDK

Enfin, toujours dans le but d'homogénéiser le développement, des layouts sont proposés comme, par exemple, les éléments simples des listes sur une ou deux lignes. Dans l'exemple 5-9 de ce chapitre, Ressources Android , nous utilisons des icônes de menu et un layout pour une liste à choix unique.

Activité construite avec des ressources standard

3. Ressources non structurées

Dans ce dossier, nous pouvons mettre tout ce qui n'est pas supporté par les ressources structurées, par exemple un fichier texte, un fichier définissant un objet 3D ou encore un fichier dans un format propriétaire.

3.1. Utilisation

Ces ressources sont accessibles grâce à la classe AssetManager . Elles sont ensuite lues sous la forme de flux à décoder selon le format du fichier. Pour la lecture d'un fichier texte, dans l'exemple 5-10, Ressources nonstructurées , nous obtenons le code qui suit.

Exemple 5-18. Récupération et décodage d'une ressource de type texte

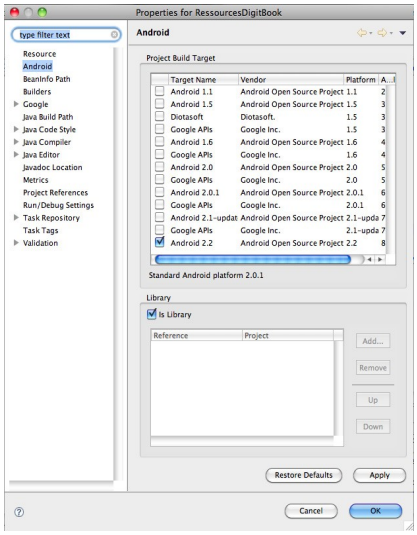
```
// Récupération de la ressource
InputStream is = getAssets().open("licence.txt");
// Récupération de la taille de la ressource
int size = is.available();
// lecture de la ressource.

byte[] buffer = new byte[size];
is.read(buffer);
is.close();

// Conversion en String.
String text = new String(buffer);
// Affichage TextView tv =
(TextView)findViewById(R.id.licence);
tv.setText(text);
```

Par contre, il faudra faire attention à la taille des fichiers car AssetManager limite la lecture à 1 Mo.

4. Création d'une bibliothèque de ressources

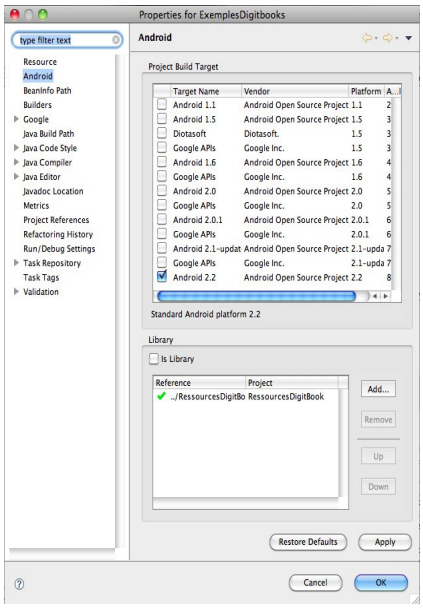
Pour améliorer la réutilisation des ressources développées, depuis la version 0.9.6 du plugin eclipse ADT , nous pouvons créer des ressources dans des projets externes. Ceci permet aussi de développer ses propres composants réutilisables. L'exemple 8, Bibliothèque de ressources , présente cette pratique en mettant en oeuvre une activité ressource dans le fichier AndroidManifest.xml présenté à l'exemple 5-20. Une fois le

projet de ressources créé (dans notre cas RessourcesDigitBooks), nous devons spécifier dans les propriétés du projet qu'il s'agit d'une bibliothèque.



Case à cocher pour faire d'un projet une bibliothèque

Ensuite, dans notre projet d'application, nous ajoutons la bibliothèque, ainsi que le dossier src, comme dossier source dans le build path.



Ajout d'une dépendance à un projet de ressources

Concernant l'accès aux ressources, il n'y a aucun changement puisque les classes R du projet et de ces bibliothèques sont fusionnées. Ainsi, il faut penser que le projet d'application écrasera toutes les valeurs de ressources si elles sont écrites en double.

Nous pouvons d'ailleurs définir l'ordre des projets de ressources pour gérer leur écrasement. Pour le code Java, il faudra importer les classes depuis le package de la bibliothèque.

Exemple 5-19. Utilisation d'une activité définie dans les ressources

```
<activity
android:name="fr.digitbooks.android.ressources.
RessourcesActivity" />
```

Ces cinq premiers chapitres terminés, nous sommes désormais armés pour réaliser les interfaces de différents types d'applications tout en supportant la variété infinie des périphériques Android. Nous allons donc maintenant aborder des parties du SDK plus fonctionnelles, comme le stockage, les appels de web services ou communications réseaux, la 3D ou encore le développement de code natif.

Retrouvez l'article de Cyril Mottier et Ludovic Perrier en ligne : Lien 19

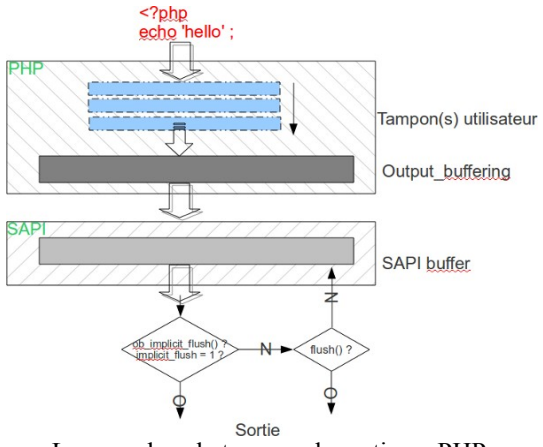
Utilisation du tampon de sortie en PHP

Lors du déclenchement d'un affichage en PHP (echo, var_dump, printf ou toute autre fonction), la chaine à afficher ne part pas directement vers l'affichage.
Elle est en réalité stockée dans différentes piles appelées "tampons", sur lesquelles l'utilisateur a un contrôle plus ou moins fin.
Lorsque le dernier tampon tout en bas est vidé, l'affichage est alors envoyé à un endroit, en fonction de la SAPI utilisée. Par exemple pour CLI, il s'agira de la sortie standard: la chaine est affichée à l'écran.
Nous allons ici détailler les différentes couches de tampon, leur utilisation et leur impact sur le code PHP.

1. Présentation du système de tampon

1.1. Les différentes couches de tampon

PHP dispose de 2 couches de tampons, dont une dite "utilisateur" (sur laquelle l'utilisateur possède beaucoup d'options), celle-ci se situe en haut de la pile.
Vient ensuite le tampon de sortie, le fameux "output_buffering" que l'on configure via un paramètre dans php.ini, c'est la couche basse de PHP
Une fois cette couche traversée, PHP va écrire dans la SAPI, or celle-ci aussi peut être bufferisée. PHP vous offre alors la possibilité éventuelle de la vider ("flush").



Les couches de tampon de sortie en PHP

1.2. Le tampon de sortie

Le tampon de sortie, appelé "output_buffering", met en tampon tout ce qui part vers l'affichage (vers la sortie standard) depuis PHP. Cette couche est configurable via le paramètre "*output_buffering*" de php.ini.
Ce paramètre peut prendre les valeurs "on", "off" ou encore un entier représentant la taille du tampon en octets. Si "on" est choisie, la taille de 40960 octets (40Ko) est utilisée.

Comme on peut s'y attendre, si la valeur "off" est utilisée, toute sortie PHP sera directement liée à l'entrée de la SAPI. Ceci n'est pas recommandé si la SAPI Apache est utilisée (mode web le plus courant) car de petites écritures répétées peuvent alors affecter les performances. La même recommandation est donnée pour CGI.

La SAPI CLI (mode ligne de commandes) désactive systématiquement le tampon PHP, quelle que soit la valeur donnée dans php.ini.
Ceci est plutôt logique, car en mode ligne de commande, on s'attend à ce que tout affichage déclenché en PHP soit immédiatement envoyé à l'écran.

1.2.1. Exemple

Cet exemple utilise la SAPI Apache (et ne fonctionnera pas en CLI).
Considérons que nous voulons afficher 3 caractères à la fois. Un caractère pèse un octet (par défaut, sans Unicode), il s'agit donc de régler *output_buffering* à 3 dans php.ini (ou au moyen de *php_value* dans .htaccess).

Considérant output_buffering=3 dans php.ini

```
<?php
echo 'aa'; // rien ne s'affiche à l'écran
sleep(3);
echo 'b'; // 3ème octet envoyé sur la sortie, le buffer est vidé et la chaine "aab" est affichée à l'écran
sleep(3);
echo 'c'; // rien ne s'affiche
sleep(3);

// le script se termine, PHP vide son buffer et 'c' est affiché à la suite
?>
```

Relativement simple.
Une valeur de 40960 (comme celle par défaut) n'est pas mauvaise dans un cas web réel. Gardez en tête qu'il vaut mieux que PHP envoie tout d'un coup vers la SAPI plutôt que morceaux par morceaux, pour des raisons de performances.

Si vous n'obtenez pas ces résultats à l'écran et que tout s'affiche d'un coup, c'est tout simplement qu'il reste encore un tampon: celui de la SAPI. Il fait l'objet du chapitre suivant.
Il est important de garder en tête que d'autres tampons peuvent encore intervenir, et ceux-là, vous n'y avez pas accès au moyen de PHP. On pourrait parler par exemple du tampon réseau (TCP buffer), ou encore d'un ou plusieurs tampons dans le noyau de l'OS. Enfin le client web, le

navigateur, peut lui aussi recevoir des infos mais ne pas déclencher l'affichage immédiatement: Là encore, PHP n'a rien à voir.

1.2.2. Les en-têtes HTTP

PHP est un langage orienté web. Ainsi, le contexte HTTP est très souvent utilisé et nécessite l'envoi des en-têtes HTTP avant le contenu. Ainsi, vous devez appeler des fonctions comme ***header()***, ***session_start()*** ou encore ***setcookie()*** avant tout déclenchement d'affichage. Cependant, si vous activez le buffer de sortie de PHP, celui-ci va non seulement stocker le contenu de la sortie, mais aussi les en-têtes que vous envoyez. Lors de sa vidange, il va lui-même envoyer d'abord les en-têtes, puis le contenu. Il va donc vous autoriser ce genre de fantaisie:

Le tampon de sortie envoie d'abord les en-têtes HTTP puis le contenu

```
<?php
/*
Nous supposons que la taille du tampon de sortie est supérieure à
9 octets (nous allons afficher 9 octets)
*/

echo "foo";
session_start();
echo "bar";
header("X-Some-Header: foobar");
echo "baz";
```

Faire des entorses à HTTP est peu recommandé, mais si on s'y connait bien on peut utiliser le tampon à notre avantage pour résoudre certaines problématiques.
Attention, dès que le tampon est vidé, PHP envoie les en-têtes HTTP et on ne pourra donc plus en envoyer par la suite.

Supposant un output_buffering=3

```
<?php
echo 'aa';
header('Foo: bar');
sleep(3);
echo 'b'; // Vide le tampon car il fait 3 octets
header('Bar: baz'); // Warning: Cannot modify header information - headers already sent by ...
```

Les en-têtes HTTP font partie de la sortie standard, ils empruntent donc le même canal que la texte affiché. La couche de tampon de PHP va simplement les séparer en interne, mais lors de l'envoi, en-têtes et contenu textuel seront envoyés dans le même tuyau (stdout).

1.3. Le tampon de la SAPI

Sous le tampon PHP se trouve le tampon de la SAPI. En effet, lorsque PHP vide son buffer, il le vide dans la SAPI qui elle aussi possède un tampon dont la taille n'est pas configurable via PHP.
Ce tampon-là peut encore retenir les informations avant de les envoyer réellement vers l'affichage (ou le client web, ou plus précisément la pile logicielle du dessous), c'est souvent le cas pour Apache (quelques octets).

Là où PHP va pouvoir prendre la main, c'est sur sa vidange: il est possible de demander à PHP de demander à la couche SAPI de vider ("flusher") son tampon. Ceci se fait de plusieurs manières en PHP:

1. manuellement, en appelant la fonction PHP ***flush()*** ;
2. systématiquement, en appelant la fonction PHP ***ob_implicit_flush()*** ;
3. systématiquement, en indiquant dans php.ini *implicit_flush=1*.

Ainsi, si Apache vous gène en gardant dans son buffer des informations, vous pouvez corriger les scripts vus dans le chapitre précédant de plusieurs manières:

Considérant output_buffering=3 dans php.ini, vidange manuelle du tampon de SAPI

```
<?php
echo 'aa';
sleep(3); // rien ne s'affiche à l'écran
echo 'b'; // 3ème octet envoyé sur la sortie, le buffer est vidé dans la SAPI
flush(); // Demande à la SAPI de vider son buffer, si aucun autre tampon n'intervient, la chaine "aab" va s'afficher
sleep(3);
echo 'c'; // rien ne s'affiche
sleep(3);
?>
// le script se termine, PHP vide son buffer ET vide aussi celui de la SAPI. 'c' s'affiche donc à la suite
```

Considérant output_buffering=3 dans php.ini, vidange automatique du tampon de SAPI

```
<?php
ob_implicit_flush(1); // Dès que le buffer de la SAPI reçoit des données, demande sa vidange immédiate
echo 'aa';
sleep(3); // rien ne s'affiche à l'écran
echo 'b'; // 3ème octet envoyé sur la sortie, le buffer est vidé dans la SAPI qui se vide automatiquement

// etc.
```

Considérant output_buffering=3 ET implicit_flush=1 dans php.ini

```
<?php
// implicit_flush est à 1 dans php.ini, dès que le buffer de la SAPI reçoit des données, sa vidange est immédiate
echo 'aa';
sleep(3); // rien ne s'affiche à l'écran
echo 'b'; // 3ème octet envoyé sur la sortie, le buffer est vidé dans la SAPI qui se vide automatiquement

// etc.
```

Dès qu'on a compris comment les couches sont empilées et comment prendre la main dessus, tout est plus clair non?

Une fois de plus, la SAPI CLI (ligne de commande) fait parler d'elle. Non seulement elle impose *output_buffering=0*, mais en plus elle impose *implicit_flush* à 1. Logique une fois de plus, tout affichage PHP en ligne de commande est immédiatement envoyé vers la console (***fflush()*** est appelée, pour les connaisseurs).

Pour plus d'informations sur le fonctionnement des SAPI, lisez SAPI et modes de communication (Lien 20).

2. Utiliser les tampons utilisateur

Au dessus du tampon PHP se trouvent les tampons utilisateur. Le développeur a pleinement la main dessus et peut en empiler autant qu'il souhaite.
Les tampons utilisateur se pilotent principalement au moyen des fonctions PHP ***ob_*()***, le manuel officiel se situant ici : Lien 21.

La taille des tampons est configurable, mais les tampons sont comme tout: des ressources, et à ce titre ils consomment de la mémoire. Plus vous utilisez de couches de tampon, à fortiori de grosse taille, plus la consommation mémoire de PHP augmentera. En général, on en manipule un seul (si on utilise cette fonctionnalité) ou 2, rarement plus.

Voyons un exemple concret d'un tampon utilisateur.

Pour plus de "réalisme", vous comprendrez maintenant pourquoi je suggère pour les exemples de ce chapitre de mettre *output_buffering = off* et *implicit_flush = 1* non ?
Je veux tout simplement que tout ce qui va sortir à un moment donné de mes tampons utilisateur soit directement envoyé sur la sortie, sans passer par la couche tampon de PHP ni tampon de la SAPI.
Si vous voulez vous simplifier la vie, testez ces exemples sur la SAPI CLI qui possède déjà les bonnes valeurs de ces 2 directives (forcées). Utilisez le mode interactif (php -a) pour plus de réactivité.

Exemple très simple de tampon utilisateur

```
<?php
ob_start();
echo "hello world"; // rien ne s'affiche
?>
// Le script se termine, et "hello world" s'affiche
```

En fait, avec *output_buffering = 0* et *implicit_flush = 1*, le fait de démarrer un tampon utilisateur avec ***ob_start()*** vous laisse le contrôle total sur ce tampon.
Pour configurer sa taille (6144 octets par défaut), utilisez le deuxième paramètre de ***ob_start()*** :

Exemple très simple de tampon utilisateur d'une taille de 3 octets

```
<?php
ob_start(null, 3); // 3 octets de tampon
echo "fo"; // rien ne s'affiche, le tampon n'est pas plein

echo "o"; // le tampon est plein et se vide, "foo" s'affiche

/*
Cette chaine est supérieure à 3 octets, elle entre et ressort
tout de suite du tampon, elle est donc affichée instantanément
*/
echo "hello world";
```

Il est possible d'empiler plusieurs tampons qui seront alors "pipés" : dès que celui du dessus se vide, il remplit celui du dessous, et ainsi de suite. On peut donc obtenir des résultats rigolos, voici un exemple :

Empiler plusieurs tampons

```
<?php
ob_start(null, 10); // tampon 1, en bas de pile, taille 10 octets
ob_start(null, 8); // tampon 2, en haut de pile, taille 8 octets

echo "hello"; // remplit le tampon 2 de 5 octets (5/8)

echo "wo"; // rajoute 2 octets dans le tampon 2 (7/8)

/*
rajoute 2 octets dans le tampon 2 (9/8), le tampon est vidé dans celui du dessous, tampon 1 est à 9/10
donc toujours rien n'est affiché. Tampon 2 est à 0/8
*/
echo "rl";

/*
rajoute 10 octets dans tampon 2 (10/8), le tampon est vidé dans celui du dessous
tampon 1 est rempli par 10 octets, il est donc à 19/10 et se vide, comme plus rien
n'est en dessous, la chaine "hello world, comment" s'affiche
*/
echo "d, comment";

echo "ca va?" // rajoute 6 octets dans tampon 2 (6/8)

?>
/*
Le script se termine, PHP dépile les tampons et ce qui reste fini par s'afficher: "ca va?"
*/
```

On ne peut QUE remplir le tampon du haut de la pile, tous ceux plus bas sont intouchables tant que celui du haut n'est pas détruit (on va y venir).
On ne peut toucher à l'ordre de la pile, mais on peut prendre connaissance de son état (on va y venir).

2.1. Jouer avec le contenu des tampons

Les fonctions de PHP concernant les tampons utilisateur permettent pas mal de fantaisies :

Manipuler le contenu des tampons

```
<?php
ob_start(null, 20); // tampon 1
ob_start(null, 8); // tampon 2

/*
ob_get_level() retourne le nombre de tampons actifs empilés,
dans notre cas: 2
*/
printf("%d: dvp", ob_get_level());

// A ce stade, tampon 2 contient "2: dvp" (6/8)
```

```
/*
récupère le contenu du tampon courant (haut de
pile, tampon 2)
sans pour autant le vider, ni le détruire
*/
$dvp = ob_get_contents(); // $dvp contient "2:
dvp"

echo "\nfoo"; // Le tampon actif déborde (10/8)
et se vide dans le tampon du dessous

echo "barbaz"; // tampon 2 est à 6/8, tampon 1
toujours à 10/20

/*
détruit le tampon de haut de pile (tampon 2) en
détruisant aussi
son contenu qui est alors perdu
*/
ob_end_clean();

/*
Ici il n'y a plus qu'un tampon actif, le tampon
1, tampon 2 a été détruit
et la mémoire qu'il consommait (6 octets) a été
libérée
*/

$foo = ob_get_clean(); // récupère le contenu du
tampon courant et détruit le du tampon (0/20)

echo "booboo";

ob_end_flush(); // Vidange le tampon courant et
détruit le, "2: dvp\nfoobooboo" est perdu
définitivement

// A ce stade, plus aucun tampon utilisateur
n'existe
```

Comme vous pouvez le noter, on a un contrôle fin sur les tampons, mais les fonctions ne donnent toujours accès qu'au tampon de plus haut niveau.
On peut encore remettre à zéro le contenu du tampon courant (donc supprimer son contenu) avec ***ob_clean()***, le vidanger (le purger vers le buffer au niveau juste en dessous) avec ***ob_flush()***, ou encore obtenir la taille de son contenu courant (en octets) au moyen de ***ob_get_length()***.

Si vous le souhaitez, vous pouvez verrouiller un tampon, c'est à dire empêcher toute manipulation de suppression ou de vidange. Pour cela, passez *false* au troisième paramètre de ***ob_start()*** ("erase") :

Verrouillage d'un tampon contre la vidange ou la suppression

```
<?php
ob_start(null, 1024, false);
echo "**du contenu ici** : on peut y aller, 1024
caractères sont autorisés avant vidange";

ob_flush(); // Erreur de type Notice
ob_clean(); // Idem

/*
Et idem pour ob_end_clean(), ob_end_flush(), vous
ne pouvez *plus du tout*
supprimer ce tampon, vous êtes bloqué à ce niveau
et ne pouvez descendre plus
bas.
*/
```

Attention: TOUT contenu envoyé vers l'affichage est mis en tampon. Une notice PHP, si *display_errors* est à On dans php.ini, va non seulement s'afficher en évitant tous les tampons, mais en plus aller se loger dans le tampon courant (et peut le faire déborder).
Ce choix d'implémentation est important à noter (les développeurs de PHP ont considérés qu'une erreur devait absolument être affichée quel que soit le nombre de tampons actifs sur le moment).
Il reste possible si on ne veut pas du tout afficher l'erreur, de rediriger *display_errors* sur stderr ou de mettre le paramètre à off tout simplement.

Attention aux erreurs (non fatales) de PHP!

```
<?php
ob_start(); // tampon de 6144 octets par défaut
echo $a; // Une erreur manifeste de type "Notice"

/*
Le message d'erreur s'affiche à l'écran en
évitant tous les tampons
mais il a tout de même été stocké dans le tampon
courant

PHP Notice:  Undefined variable: a in php shell
code on line 1
PHP Stack trace:
PHP   1. {main}() php shell code:0
*/

$error = ob_get_contents();
ob_end_clean(); // Détruit le tampon courant et
son contenu

echo $error;
/*
L'erreur précédente s'affiche ici, elle était
bien stockée dans
le tampon et a été récupérée dans $error
*/
```

2.2. Les callbacks de buffer

Dans notre utilisation précédente de la fonction ***ob_start()***, nous avions toujours passé son premier paramètre à *null*. Celui-ci permet d'indiquer une fonction de callback PHP à exécuter une fois la vidange ou le nettoyage du tampon.
La fonction reçoit comme paramètre une chaîne représentant le contenu du tampon courant ainsi qu'une constante d'état, et elle doit renvoyer une chaîne de caractères. Si la fonction retourne *false*, alors le tampon n'est pas traité et est envoyé, comme si la callback était transparente (elle aurait tout aussi pu renvoyer le contenu qu'elle a reçu).

Exemple de fonction de callback sur un tampon utilisateur

```
<?php
function dvp_callback($bufferContent)
{
    return sprintf('<div class="content">
%s</div>', $bufferContent);
}

ob_start('dvp_callback');
echo "hello world";

/*
Le script PHP se termine, les tampons sont vidés
```

```
et la callback est exécutée.
"<div class="content">hello world</div>"
s'affiche
*/
```

Comme on peut s'y attendre, la fonction de callback doit exister et doit être invocable, ne doit pas lever d'exception, ne doit pas déclencher d'affichage et ne doit pas piloter le tampon en appelant ***ob_flush();*** ou autres fonctions.
Il n'est jamais bon de faire tourner PHP en bourrique. Tous ces scénarios ont été prévus, certes, mais ne sont pas recommandés et peuvent mener à des comportements indéfinis.

La fonction de callback a aussi moyen de détecter 2 situations :

1. si une fonction comme ***ob_flush()*** ou ***ob_clean()*** a été invoquée (opération sur le tampon mais sans le fermer) ;
2. si la fermeture du tampon a été invoquée ("end").

Elle reçoit la situation en cours en second paramètre. Ce paramètre est un masque entre *PHP_OUTPUT_HANDLER_START* (1), *PHP_OUTPUT_HANDLER_CONT* (2) et *PHP_OUTPUT_HANDLER_END* (3).
Cela peut être pratique pour prendre des décisions dans des cas précis.

La fonction de callback sait quand elle est appelée

```
<?php
$bufferOperations = 0;

function dvp_callback($bufferContent, $mode)
{
    global $bufferOperations;

        /*
      Si une opération autre que la fermeture du
tampon a été effectuée,
      la compter. Il peut s'agir soit d'un
ob_clean(), soit d'un ob_flush(),
      on ne peut le savoir précisément.
    */
    if ($mode & PHP_OUTPUT_HANDLER_CONT) {
            $bufferOperations++;
    }
    if ($mode & PHP_OUTPUT_HANDLER_END) {
        // Ici la fermeture du tampon a été
commandée, ob_end_flush() ou ob_end_clean();
    }
    return false; // Retourne le tampon comme il
arrive
}

ob_start('dvp_callback');
echo "hello world";
ob_flush(); // "hello world" s'affiche
ob_end_clean();

echo $bufferOperations; // 1
```

Il est possible d'affecter une fonction de callback au tampon de sortie natif de PHP (chapitre I-B). Pour cela, indiquez son nom dans la directive *output_handler* de php.ini, en activant bien sûr le tampon de sortie PHP au moyen de *output_buffering* dans php.ini (une valeur autre que Off ou 0).

3. Les bonus

Ils sont documentés, mais bon...
output_add_rewrite_var() , je vais vous laisser lire la documentation (Lien 22), nul besoin de la répéter.
Cette fonction est intéressante car elle va elle-même démarrer un tampon en lui ajoutant une callback maison (nommée "URL-Rewriter") réécrivant les liens et les formulaires.

Concernant les fonctions de callbacks que PHP fournit (au moyen d'extensions, à activer), on peut citer :

1. ob_gzhandler() : Gère la compression Gzip (conflit avec ob_inflate/deflatehandler()) ;
2. mb_output_handler() : Gère des conversions de jeu de caractères en sortie ;
3. ob_iconv_handler() : Gère des conversions de jeu de caractères en sortie ;
4. ob_tidyhandler() : Gère l'analyse syntaxique du HTML de sortie pour réparations diverses ;
5. ob_[inflate/deflate]handler() : Gère la compression Gzip (conflit avec ob_gzhandler()) ;
6. ob_etaghandler() : Gère les en-têtes de cache HTTP "Etag" en sortie.

Certaines sont vraiment très efficaces, je pense à ***ob_iconv_handler()*** ou encore ***ob_tidyhandler()***, essayez si vous ne connaissez pas.

Rappel: Si vous aviez codé des callbacks en PHP faisant le même travail que celles présentées dans la liste ci-dessus, utilisez plutôt celles de la liste: elles sont écrites en C et seront beaucoup plus rapides qu'une même implémentation écrite en PHP.

ob_list_handlers() retourne un tableau contenant les noms de callbacks en cours d'utilisation dans les éventuels différents tampons utilisateur ET éventuellement le tampon PHP.
Puis plutôt intéressante, ***ob_get_status()*** donne plein d'informations sur les tampons en cours d'utilisation.

Statistiques sur les tampons

```
<?php
ob_start();
echo "foo";
ob_start(function ($content) { return false; },
200, false);
ob_start(null, 400);

$infos = ob_get_status(1); // Mettre le premier
paramètre à true (ou 1)
for ($i = 0; $i < ob_get_level(); $i++) {
    ob_end_clean();
}

var_dump($infos);
/*
array(3) {
  [0]=>      *buffer 0, tout en bas*
  array(7) {
    ["chunk_size"]=>
    int(0)
    ["size"]=>
    int(40960) *40Ko*
```

```
    ["block_size"]=>
    int(10240)    * Allocations mémoires de 10Ko
(segmentation) *
    ["type"]=>
    int(1)
    ["status"]=>
    int(0)
    ["name"]=>
    string(22) "default output handler"
    ["del"]=>
    bool(true)    * non verrouillé *
  }
  [1]=>
  array(5) {
    ["chunk_size"]=>
    int(200)
    ["type"]=>
    int(1)
    ["status"]=>
    int(0)
    ["name"]=>
    string(17) "Closure::__invoke"
    ["del"]=>
    bool(false)    * verrouillé *
  }
  [2]=>
  array(5) {
    ["chunk_size"]=>
    int(400)
    ["type"]=>
    int(1)
    ["status"]=>
    int(0)
    ["name"]=>
    string(22) "default output handler"
    ["del"]=>
    bool(true)
  }
}
*/
```

4. Cas pratiques

A part les classiques (capturer la sortie de ***var_dump()*** ou de toute autre fonction), je n'en ai pas personnellement, mais on en trouve sur le web. Par exemple dans des scripts en ligne de commande écrits en PHP.

Imaginez un programme PHP qui doit télécharger un fichier en ligne, puis le copier sur le disque. Il pourrait être intéressant d'afficher une barre de progression du téléchargement (comme wget par exemple). Afficher peu à peu "=>" "==>" "===>" etc. pourrait être effectué avec un contrôle fin d'un tampon de sortie.
On peut extrapoler au web pour les mêmes cas, ou encore pour commencer à afficher du contenu à un internaute impatient, alors que l'on calcule encore un résultat (en général on fera plutôt l'inverse, c'est à dire allouer les ressources lourdes -comme une base de données- en début de script, pour finalement tout afficher d'un coup).

Coté Zend Framework, les tampons sont utilisés dans plusieurs cas pratiques. Regardez ainsi du coté de **Zend_Controller_Response_Http**, **Zend_Controller_Dispatcher_Standard**, **Zend_Soap_Server**, **Zend_View_Helper_PlaceHolder** et sa fameuse méthode ***captureStart()***, etc.

5. Conclusion

Maîtriser la manière dont PHP "s'occupe de ce qu'il lui est demandé d'afficher" permet de comprendre un aspect du langage et certains comportements qu'il peut avoir que l'on juge souvent "bizarres".
Les tampons de sortie sont présents dans presque toutes les couches logicielles, afin d'éviter trop de "petites" communications trop souvent, souvent couteuses en performances notamment processeur. Lorsqu'il s'agit du réseau, TCP/IP fait un travail remarquable à ce sujet, et certains matériels réseaux ne se privent pas de mettre en tampon tout un tas de paquets avant de les expédier sur une autre interface pour éviter de surcharger le réseau.
Je m'écarte un peu non ?

J'espère que cet article aura fait germer des idées dans certaines têtes, ou au moins éclairci certains points sur une fonctionnalité de PHP souvent peu connue.

SAPI et modes de communication en PHP : Lien 23
PHP Internals : fonctionnement global de PHP : Lien 24

Retrouvez l'article de Julien Pauli en ligne : Lien 25

Opérations booléennes et logique binaire en PHP

Savoir manipuler des données binaires en base 2 ou en base 16 (hexadécimal) peut s'avérer nécessaire ne serait-ce que pour la compréhension d'un algorithme.
PHP propose pas mal de fonctionnalités à ce sujet, nous allons les passer en revue avec quelques exemples concrets.

1. Introduction

S'il est un sujet que peu de développeurs PHP connaissent, c'est bien celui des mathématiques binaires et des opérations booléennes. C'est étonnant, car le processeur de toute machine ne pratique que ça.
Certes PHP est un langage plutôt orienté développement web, et le web n'a pas fondamentalement besoin que l'on connaisse ces principes. Mais PHP est un mortier qui sait faire beaucoup de choses, toujours est-il que tout est binaire autour de nous en informatique, et qu'à un moment ou un autre, vous aurez besoin de comprendre cette manière de voir les choses pour effectuer tout un tas d'opérations.

2. Rappels sur la représentation binaire et les bases

Les bases, en mathématiques, sont des manières de voir les choses. Nous comptons (nous humains) en base 10. Voyons un exemple.

La base 10, nous y sommes extrêmement habitués

```
Je pense au hasard au nombre 12974. Et bien
celui-ci, exprimé en base10 est décomposable en
somme de puissances de dix :
12974 = 4*10^0 + 7*10^1 + 9*10^2 + 2*10^3 +
1*10^4
```

Tout le monde est d'accord, impossible de me contredire ici.
La machine, elle, compte en base 2, car l'électronique sur laquelle elle est basée, gère extrêmement bien deux états: allumé-On-1 contre éteint-Off-0, au moyen de composants appelés transistors, qui pour rappels se trouvent par milliards dans les microprocesseurs modernes.
Ainsi, si je reprends la même logique avec 12974 en base 2:

La base 2, les ordinateurs n'utilisent que ça

```
12974 peut aussi s'exprimer en base 2 : c'est
alors une somme de puissances de 2
12974 = 1*2^12 + 1*2^11 + 1*2^9 + 1*2^7 + 1*2^5 +
1*2^3 + 1*2^2 + 1*2^1
```

Oui bon, la base est plus petite, ceci se traduit par moins de chiffres disponibles (0 ou 1 au lieu de 0 ou 1 ou 2 ou... 9) mais en contrepartie plus de bits.
Pour creuser (convertir de base en base par exemple), voyez Wikipédia : Lien 26.

En fait, la base 2 est plus simple que la base 10 (vous ne le croyez pas car votre cerveau est trop habitué à la base 10, c'est tout) mais il faut plus de bits pour représenter le même chiffre au final.
Tout est question de puissance, et de puissances de 2. Le processeur traite donc des opérations extraordinairement simples : elles ne peuvent être composées que de "0" ou de "1" ; et c'est pour cela qu'il en traite des millions/milliards par seconde.

2.1. Rappels sur les bits, les octets et le vocabulaire

Un bit est une unité logique d'information. Il vaut 0 ou 1 en base 2. Un octet est l'association de 8 bits et s'écrit en anglais *'one byte'*, attention ! (un bit en anglais se dit *'one digit'*). Un processeur, aujourd'hui, sait effectuer des calculs divers et variés, mais ses opérandes ne peuvent dépasser le maximum de 32 bits (ou plutôt 64 de nos jours, tout de même) soit 4 octets.
Ainsi, toute valeur numérique supérieure au maximum que l'on peut stocker sur 32 bits (ou 64) n'est pas représentable de manière simple. Ainsi, un entier ne peut dépasser (2^32)-1 soit 4294967295 non signé (32bits).
32 bits, c'est long à représenter et 64 encore plus. On a donc souvent recours à la base 16: l'hexadécimal. La base 16 utilise 16 chiffres mais notre numération ne sait en représenter que 10: de zéro à neuf. Pour aller de 10 à 16, on utilise donc les lettres de A à F.

Encore un peu de vocabulaire, on appelle un Mot (Word) un ensemble de 16 bits (2 octets). Un Double-mot (DWORD) 32 bits et un Quatrain (QWORD) pour 64 bits. Il existe aussi le DQWORD.
En théorie, un mot doit correspondre à la longueur du bus du processeur (normalement, aujourd'hui, un mot devrait mesurer 64 bits). Mais pour des raisons historiques, le mot pèse 16 bits.

valeurs décimales	128	64	32	16	8	4	2	1
valeurs (base 2)	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
bits	0	1	0	1	1	0	0	1

Calcul avec des octets en base 2, ici la valeur décimale 89 est représentée

valeurs décimales	4096	256	16	1
valeurs (base 16)	16^3	16^2	16^1	16^0
bits	0	0	5	9

Calcul avec des octets en base 16, ici la valeur décimale 89 est représentée

89 en base 10 = 01011001 en base 2 ou encore 59 en base 16 aussi écrit 0x59 (l'hexadécimal est préfixé par 0x lorsqu'on l'écrit, par convention pour reconnaitre la base 16).

Reste encore la base octale: la base 8. Lorsqu'on fait ce genre de calcul souvent (langage C, assembleur ou encore électronique numérique), on arrive à convertir les bases de tête de manière très rapide (au moins jusqu'à l'octet voire jusqu'au mot). Tout est question de division/multiplication, de retenue... de mathématiques très simples.

Il reste possible pour les débutants d'utiliser des calculatrices gérant les bases. *Gcalc* est très bien dans le genre.

Dernière remarque : l'ordre des bits. On appelle **le bit de poids faible** celui qui possède la puissance la plus basse, inversement pour **le bit de poids fort**. Soit notre 89 décimal. Il donne en binaire 01011001 ou ... 10011010 ?? Dois-je exprimer le bit de poids faible en premier, ou bien le bit de poids fort ? Quelle question et quel casse-tête informatique, surtout lorsque les constructeurs (et les OS) font ce qu'ils veulent ! Le fait de préciser le bit de poids faible en premier est appelé l'ordre **Little Endian**, au contraire, présenter le bit de poids fort en premier représente l'ordre **Big Endian**.

3. Opérations binaires en PHP

PHP permet d'effectuer les opérations binaires au moyen des opérateurs sur les bits : Lien 27. Il possède aussi quelques fonctions qui manipulent les bases.
Les opérations logiques OR, AND, NOT et XOR sont toutes possibles via les opérateurs de bits. Attention, tout

ceci suppose systématiquement le type PHP integer (exprimé souvent en hexadécimal); un transtypage sera effectué dans le cas contraire (sauf pour les chaines, PHP utilisera alors la table ASCII pour la conversion vers l'entier).

3.1. OR, ou l'affectation d'un bit

L'opération OR met les bits de la sortie à 1 qui sont à 1 dans une des deux opérandes. Cette opération se fait au moyen de '|' en PHP :

Exemple d'un OR logique en PHP

```
<?php
$a = 0xA3; // 1010.0011
$b = 0x3C; // 0011.1100

$c = $a | $b; // 1011.1111 soit 0xBF ou encore 191 en décimal
```

$a	1	0	1	0	0	0	1	1
$b	0	0	1	1	1	1	0	0
Résultat OR	1	0	1	1	1	1	1	1

OR

Nous venons de manipuler des octets. Rappelez-vous qu'un octet représente 8 bits. Pour mieux les calculer, je les représente avec un point au milieu: '1001.0110', c'est juste pour la lisibilité et l'aide au calcul mental.
Un octet se représente donc en hexadécimal avec 2 digits: 0x00 est 0 en décimal et 0xFF est 255 en décimal.

Ne confondez surtout pas l'opérateur '|' et l'opérateur '||'. L'opérateur '|' agit sur les bits d'un entier, alors que '||' convertira les opérandes en booléen, rien à voir (en fait si, un booléen n'est rien d'autre qu'un et un seul bit : vrai (1) ou faux (0). On ne va pas aller loin avec un seul bit...).

L'opérateur OR est surtout utilisé pour affecter les bits dans un entier :

Utilisation du OR logique pour affecter les 4 bits de gauche

```
<?php
$a = 0x07;    // 0000.0111

// Met les 4 bits de gauche à 1 dans $a
$a |= 0xF0;   // 0000.0111 OR 1111.0000 = 1111.0111

// $a vaut ici 0xF7
```

L'opération |= met à 1 les bits de l'opérande de gauche étant à 1 dans l'opérande de droite. Pour les mettre plutôt à 0, il faut utiliser l'opérateur XOR (OU exclusif).

3.2. AND, ou le masque de bits

L'opération AND met les bits de la sortie à 1 qui sont à 1 dans les deux opérandes en même temps. Cette opération se fait au moyen de '&' en PHP :

Exemple d'un AND logique en PHP

```
<?php
$a = 0xA3; // 1010.0011
$b = 0x3C; // 0011.1100

$c = $a & $b; // 0010.0000 soit 0x20 ou encore 32 en décimal
```

$a	1	0	1	0	0	0	1	1
$b	0	0	1	1	1	1	0	0
Résultat AND	0	0	1	0	0	0	0	0

AND

Ne confondez surtout pas l'opérateur '&' et l'opérateur '&&'. L'opérateur '&' agit sur les bits d'un entier, alors que '&&' convertira les opérandes en booléen, rien à voir (en fait si, un booléen n'est rien d'autre qu'un et un seul bit : vrai (1) ou faux (0). On ne va pas aller loin avec un seul bit...).

L'opérateur AND est surtout utilisé pour tester les bits dans un entier, on parle de **masque de bits** :

Utilisation du AND logique pour créer un masque de bits

```
<?php
$a = 0x07;    // 0000.0111
$a &= 0xF0;   // 0000.0111 AND 1111.0000 = 0000.0000

// $a vaut ici 0x00
```

L'opération &= met à 1 les bits de l'opérande de gauche étant à 1 dans l'opérande de gauche ET de droite. Cela permet un masque, c'est-à-dire une sélection fine des bits sur lesquels on veut travailler.
En faisant un AND avec 0xF0, on veut travailler les 4 bits les plus à gauche de l'opérande, inversement en faisant un AND avec 0x0F, on veut travailler sur les 4 bits les plus à droite de l'opérande (et enfin un AND avec 0x3C permettra de travailler les bits 'du milieu', c'est moins commun).
Ceci est une technique fondamentale de l'informatique générale dans le traitement de l'information. Tous les processeurs et tous les programmes informatiques sans exception ont recours à un moment donné au masquage de bits, en général sur une donnée d'une longueur de 32bits (ou 64 de nos jours).
Exemple en l'air: que vous inspire la notion de 'masque de sous-réseau' ? Si vous n'avez jamais creusé cette notion, filez-y vite !

3.3. Shifting, ou les décalages des bits

Lorsqu'on travaille sur les bits, on a vu qu'on pouvait - dans un octet de 8 bits par exemple - travailler sur n'importe quel bit, simplement grâce à deux opérations élémentaires : AND et OR (restent XOR et NOT encore).
L'opération OR permet de mettre à 1 n'importe quel bit, l'opération XOR met à zéro n'importe quel bit, enfin l'opération AND permet de sélectionner par masquage certains bits dans le but de travailler avec.

OR, AND et NOT sont les trois opérations élémentaires fondamentales de l'algèbre de Boole : Lien 28. Il en existe d'autres remarquables qui en découlent (XOR par exemple).
Mais on peut encore s'amuser avec les bits d'une autre manière : en les décalant sur la gauche ou la droite.
L'opérateur de décalage gauche '<<', décale les bits à

gauche de N rangs, déterminés par l'opérande :

Exemple de décalage de bits à gauche

```
<?php
$a = 0x26; // 0010.0110

// Décale les bits de $a de 2 rangs vers la
gauche
$a <<= 2 // 1001.1000 soit 0x98
```

L'opérateur de décalage droite '>>', décale les bits à droite de N rangs, déterminés par l'opérande :

Exemple de décalage de bits à droite

```
<?php
$a = 0x26; // 0010.0110

// Décale les bits de $a de 2 rangs vers la
droite
$a >>= 2 // 0000.1001 soit 0x09
```

Remarquez comme les bits sont comblés par des zéros, que ce soit à gauche, ou à droite, lorsqu'ils 'sortent' du rang. Ceci n'est pas vrai pour le bit de poids fort s'il est à 1 et décalé à droite: il est conservé.
Ceci est dû au fait que le bit de poids fort représente le signe des entiers signés, et PHP utilise des entiers signés.

Vous aurez sans doute remarqué que décaler les bits d'un rang vers la gauche, revient à multiplier la valeur décimale par 2, et inversement.
Attention, je parle de valeurs décimales non signées. Dans les valeurs signées, le bit de poids fort représente le signe, et tous les autres bits sont complémentés à 2. Je ne préfère pas m'étendre sur ce point.

Le décalage de bits va permettre de compléter l'opération de masquage (AND). En effet, dans un octet (toujours pour notre exemple), il y a 8 bits que l'on peut séparer en 2 groupes de 4 bits.
Chaque groupe peut représenter quelque chose de différent, et passer d'un groupe à l'autre est alors très simple, il suffit de décaler les bits de 4 positions :

Exemples de décalages

```
<?php
$a      = 0x94; // 1001.0100
$masque = 0x5; //0101

$bitsDeGauche = $a >> 4 // 0000.1001 ou plus
simplement 1001

$bitsDeGaucheMasques = $bitsDeGauche &
$masque; // 0001
```

Un autre exemple de fonction PHP permettant de lire les n bits d'un entier à partir de la position p. la position 0 est supposée le bit de poids faible (le plus à droite) :

Jouons avec les bits en PHP

```
<?php
/*
Lit les $n bits de $x en partant de $p
$p=0 est le bit de poids faible
*/
function readDigit($x, $p, $n)
{
    return ($x >> $p+1-$n) & ~(~0 << $n);
}

$byte = 0x9D; // 1001.1101

$digits = readDigit($byte, 5, 3); // 0000.0011
```

J'avoue qu'avoir fait pas mal de C facilite la compréhension d'une telle fonction. Expliquons : *($x >> $p+1-$n)* déplace les $n bits à partir de $p à l'extrême droite. *& ~(~0 << $n)* masque les bits, ~0 est un ensemble de 1 collés, décalés de $n bits vers la gauche, donc à droite apparaissent des 0, que nous inversons avec ~, le masque est crée.
Astucieux.

3.4. Fonctions PHP relatives aux bases

PHP propose quelques fonctions liées aux bases numériques, majoritairement de conversion. Il sait nativement gérer les bases 2, 8, 10 et 16 et propose aussi des conversions vers des bases plus exotiques.
decbin()*/*bindec(), ***dechex()*/*hexdec()***, ***decoct()*/*octdec()*** sont autant de fonctions dont les noms parlent d'eux-mêmes, la base 10 restant la référence.
base_convert() permet de convertir n'importe quel nombre de n'importe quelle base vers n'importe laquelle. Petite limite tout de même: pas plus que la base 36, car 36 = 26 lettres de l'alphabet + 10 chiffres. Après, on ne sait quoi utiliser comme caractère pour représenter un nombre dans la base.

Viennent ensuite les plus compliquées, mais aussi les plus utiles, j'ai nommé ***pack()*** et ***unpack()*** dont on reparlera un peu plus tard.
Enfin, on pourrait parler de ***printf()*** aussi. ***printf()*** n'effectue pas d'opérations sur les bits, mais permet de les représenter soit sous forme binaire, soit sous forme hexadécimale.

printf: un must-have lorsqu'on manipule des données binaires

```
<?php
$mot    = 0xF307;
printf('%b', $mot); // affiche 1111001100000111,
on voit que c'est plus lisible en mettant des '.'
hein ?
```

Attention, vous ne pouvez pas représenter du binaire en PHP. Par exemple $a = 01101 ne fonctionnera pas du tout comme vous le pensez ! En PHP, tout entier peut être exprimé soit en base 10 (classique, par défaut), soit en base octale (le faire précéder de zéro), soit en base hexadécimale (le faire précéder de 0x) et c'est tout : on ne peut pas le représenter en base 2, sous forme de "0" et de "1".

printf et autres

```
<?php
$octet = decbin(3672); // 1110.0101.1000
$hexa  = 0xF2D8;

printf('%b', $hexa); // 1111.0010.1101.1000

// Affiche en hexa sur 4 digits comblés de zéros
à gauche
printf('0x%04X', $octet); // 0x0E58
```

3.5. Utilisation des bases par PHP

Tout le monde connait le rapport d'erreur de PHP. Les constantes E_* (E_ALL, E_STRICT, E_NOTICE ...) sont des entiers qui représentent tous une puissance de 2, donc un bit particulier. Regardez plutôt leur valeur sur la documentation officielle (Lien 29) ou en les affichant.

Ils sont codés sur 16 bits, soit 2 octets. E_ALL vaut 30719 en décimal, ce qui fait 32767 - 2048. 2048 c'est E_STRICT, donc E_ALL c'est tout sauf E_STRICT.

Calcul de E_ALL

```
E_ALL = (E_ERROR | E_WARNING | E_PARSE | E_NOTICE
| E_CORE_ERROR | E_CORE_WARNING | E_COMPILE_ERROR
| E_COMPILE_WARNING |
         E_USER_ERROR | E_USER_WARNING |
E_USER_NOTICE | E_RECOVERABLE_ERROR |
E_DEPRECATED | E_USER_DEPRECATED)
```

Petit exemple :

Les masques du gestionnaire d'erreur PHP

```
<?php
$error_reporting = E_ALL | E_STRICT | ~E_NOTICE;

$error_reporting = 0x7FF | 0x800 | 0xFF7; // Soit
0xFFF ou encore 4095 en décimal
```

J'avoue que même habitué, déjà sur 2 octets c'est plus difficile à calculer de tête (encore qu'en décalant les octets et en connaissant ses tables de multiplication 2 et 16 ça va).
Donc lorsque vous affichez le rapport d'erreur en décimal, et que vous voyez par exemple 6138, avec de l'entrainement vous pourrez dire quels bits sont dedans, et donc à quelles constantes ça correspond. Reste la calculatrice sinon ! Vous pouvez aussi le demander à PHP au moyen de ***printf()***.

Les masques de bits ne sont pas utilisés que dans le rapport d'erreur de PHP, mais partout : dans PDO, dans Json, dans GD...

4. Exemple concret

4.1. Gestion de droits

Nous allons mettre en place un minisystème de gestion de droits d'accès à des ressources. Nous allons supposer quatre permissions, et deux drapeaux supplémentaires :

Liste des permissions

- permission Read ;
- permission Write ;
- permission Lock ;
- permission Delete.

Liste des drapeaux supplémentaires

1. drapeau Activated ;
2. drapeau Gold.

Nous allons coder l'ensemble de ce système sur 1 octet, donc 8 bits. Comme nous avons 6 bits à utiliser, il en restera 2 inusités pour le futur, éventuellement.

Nos constantes de droits

```
<?php
define READ   = 1; // ou 0x01 soit 0000.0001
define WRITE  = 2; // ou 0x02 ou encore 1 << 1
soit 0000.0010
define LOCK   = 4; // ou 0x04 ou encore 1 << 2
soit 0000.0100
define DELETE = 8; // ou 0xO8 ou encore 1 << 3
soit 0000.1000

define ACTIVE = 16; // ou 0x10 ou encore 1 << 4
soit 0001.0000
define GOLD   = 32; // ou 0x20 ou encore 1 << 5
soit 0010.0000

/*
N'avoir aucun droit, c'est être à 0x00
Avoir tous les droits, c'est être à Ox3F, soit
0011.1111

Les 2 bits de gauche sont inusités, donc
l'intervalle
0x40 - 0xC0 ne sera pas utilisé
*/
```

Il ne reste plus que des fonctions affectant/enlevant un droit à quelqu'un, et une fonction de vérification de ce droit pour autoriser/interdire l'accès :

Quelques fonctions simples pour la gestion des droits

```
<?php
function allow(&$registry, $access)
{
    $registry |= $access
}

function deny(&$registry, $access)
{
    $registry ^= $access
}

function isAllowed($registry, $access)
{
    return $registry & $access;
}
```

Et on l'utilise:

Exemple (totalement fictif) d'utilisation de nos fonctions

```
<?php
$membre1 = READ | WRITE | ACTIVE;
$membre2 = 0x3F; // Tous les bits à 1

$news    = READ | GOLD;

if (isAllowed($membre1, $news)) {
    echo "Welcome to news1";
}

deny($membre2, $news);
```

A chacun de creuser...

4.2. Analyse des en-têtes binaires d'une image

Dans cet exemple, nous allons faire un travail totalement binaire : nous allons décortiquer les en-têtes d'une image PNG afin d'en récupérer les métadonnées.
Pour cela, il faudra lire X octets à partir d'un certain offset

dans l'en-tête (la spécification du format est ainsi indispensable), et interpréter ces octets.

Si on lit la spécification du format PNG (Lien 30), on s'aperçoit que les octets sont rangés de la sorte :

Libellé	Taille	Remarque
Marquage universel	8 octets	Obligatoirement: (décimal)137 80 78 71 13 10 26 10
Taille des données du segment	4 octets BigEndian	un entier non signé
Type Segment	4 octets, ASCII	Le premier DOIT être 'IHDR'
Données du Segment	La taille a été précisée auparavant	Pour IHDR, on aura : width(4 octets), height(4 octets) bit depth(1 octet), color type(1 octet), compression method(1 octet), filter method(1 octet), interlace method(1 octet)

Une partie de la spécification PNG

Il ne reste plus qu'à traiter tout ça. C'est simple: on ouvre le fichier en mode binaire et on consomme les 8 + 4 + 4 + 13 = 29 premiers octets que l'on va découper et interpréter (pour rappel 29 octets c'est 29*8 soit 232 bits).

```
<?php
$fp = fopen('fichierPng', 'rb');
$header = fread($fp, 29);
fclose($fp);

// ...
```

Ici, la problématique est que *$header* est de type string, pas de type int. Or en PHP, toute manipulation sur les bits ne peut se faire que sur des entiers, si on effectue de telles opérations sur des chaines, PHP se servira de la table ASCII pour comprendre ce qu'on lui dit, octet par octet. Ce n'est pas ce que l'on veut.
Il nous faut donc une fonction qui sait analyser les octets dans une chaine binaire, et en retourner quelque chose d'utile. Accueillez à bras ouvert ***unpack()***.

unpack() est exactement la fonction qu'il nous faut : elle analyse les octets dans une chaine et les transforme au format que l'on souhaite. Voyons cela:

```
<?php
$data =
unpack('A8head/Nsize/Atype/Nwidth/Nheight/cdepth/
ccolor/ccompress/cfilter/cinterlace' ,$header);
var_dump($data);

/*
array(10) {
  ["head"]=>
  string(8) "?PNG?"
  ["size"]=>
  int(13)
  ["type"]=>
  string(1) "I"
  ["width"]=>
  int(1212436992)
  ["height"]=>
  int(8192)
  ["depth"]=>
  int(0)
  ["color"]=>
  int(0)
  ["compress"]=>
  int(32)
  ["filter"]=>
  int(8)
  ["interlace"]=>
  int(6)
}
*/
```

Tous les formats de ***unpack()*** sont dans la documentation (Lien 31), on donne des noms à chacun des index dans le tableau de résultats et on sépare chaque donnée extraite par un '/'. Le caractère '@' permet de se déplacer à l'octet numéro xxx.

Attention, méfiez-vous bien de la taille en octets ou en bits de la donnée traitée. Si vous exprimez la mauvaise taille, PHP va manger plus ou moins de données et ne sera pas en mesure d'interpréter la donnée correctement.

Pour y voir plus clair, aidez-vous d'un éditeur hexadécimal. J'utilise *hexdump -C* en ligne de commandes, sinon *ghex2* sous le bureau.

Toutes les manipulations que l'on vient de faire là ont déjà été réalisées, en C (langage beaucoup plus approprié pour cela), dans la *libpng*, utilisée par l'extension PHP *ext/gd*.
Il vaut toujours mieux se reposer sur une implémentation en C, généralement plus rapide et efficace, même si dans notre cas, l'utilisation de PHP n'est vraiment pas beaucoup plus lourde.

4.3. Echange brut des valeurs de 2 variables

Aussi appelée le "XOR_swap", cette technique basée sur l'opération OU EXCLUSIF (XOR), consiste à échanger le contenu de 2 variables distinctes, sans passer par une variable intermédiaire :

XOR Swap en PHP
```
<?php
$a = 0x6C;
$b = 0xC8;

function xor_swap(&$v1, &$v2) {
    if ($v1 != $v2) {
    $v1 ^= $v2;
    $v2 ^= $v1;
    $v1 ^= $v2;
    }
}

xor_swap($a, $b);

printf('$a vaut maintenant 0x%2X et $b vaut 0x
```

```
%2X', $a, $b);
// Affiche effectivement $a vaut 0xC8 et $b vaut
0x6C
```

Une technique plutôt utilisée en C à l'époque ou une variable intermédiaire de type int était trop grosse pour la mémoire (ou encore pour de l'embarqué où la mémoire est extrêmement limitée).
Beaucoup d'algorithmes de cryptographie utilisent des principes similaires (registres à décalage, Réseau de Feistel...). Il est vrai qu'en PHP c'est plus rare, ça va sans dire.

5. Conclusions

Savoir manipuler des bits et des octets est une base fondamentale de l'informatique. PHP n'est pas spécialement le langage adapté à ce cas, mais il propose tout de même quelques fonctions et opérateurs sympas, très largement empruntés du langage C (sur lequel il repose), maître incontestable dans ce domaine.
Comme nous avons pu le voir, même en PHP, il peut parfois être utile de manipuler des données binaires au niveau de l'octet, que ce soit pour manipuler PHP lui-même qui parfois vous le proposera (son rapport d'erreur par exemple), ou pour inspecter n'importe quel fichier binaire, au moyen des superbes fonctions ***pack()*** et ***unpack()***. Tant de cas peuvent encore se présenter ...

Le langage PHP ne doit pas faire oublier les fondamentaux de l'informatique, et je conseille à chacun ne maitrisant pas ces aspects-là de se pencher dessus à temps perdu, cela permet de comprendre les piles des systèmes d'informations qui nous entourent, quels qu'ils soient. L'application la plus visible de la manipulation de bits est sans doute la couche réseau et ses protocoles (nous aurions d'ailleurs pu prendre cela comme exemple dans cet article).

L'Algèbre de Boole : Lien 28
Mathématiques binaires appliquées : algorithme du CRC : Lien 32
Les opérateurs de bits en C : Lien 33

Retrouvez l'article de Julien Pauli en ligne : Lien 34

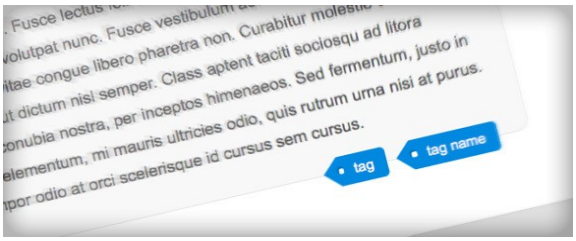
Tags pour les posts en css3

Cet article est la traduction de : Pure CSS3 Post Tags (Lien 35).
Retrouvez toutes les traductions de CSS Globe disponibles sur cssglobe.developpez.com : Lien 36.

1. Tags pour les posts en CSS3

Il s'agit d'une astuce en CSS assez simple que vous pouvez utiliser pour styler les tags de vos posts de blog, habituellement placés en bas des posts.

Les tags en CSS utilisent au moins deux astuces CSS : les triangles en CSS et les cercles en CSS.



Jetez un oeil à la démo : Lien 37

Pour apprendre à créer des triangles en CSS, je vous suggère de lire cet article : Lien 38. En bref, vous avez besoin de manipuler les bordures d'un élément dont la hauteur et la largeur sont égales à 0.
Les cercles en CSS sont plus simples à mettre en place. Tout ce dont vous avez besoin est un carré avec les coins arrondis dont la valeur est égale au moins à la moitié de la taille de l'élément. La propriété *border-radius* permettra de fusionner les coins afin d'en faire un cercle.

2. HTML

J'ai l'habitude d'utiliser un tag (balise HTML) de liste non ordonnée. Ainsi le balisage est assez simple :

```
<ul class="tags">
  <li><a href="#">tag</a></li>
  <li><a href="#">tag name</a></li>
</ul>
```

Je vais ajouter les pseudoéléments **:after** et **:before** et les styler afin d'obtenir le rendu que je veux.

3. CSS

Je place les tags de la liste en bas de l'élément post en ajustant la position du UL en absolue.

```
.tags{
        margin:0;
        padding:0;
        position:absolute;
        right:24px;
        bottom:-12px;
        list-style:none;
}
```

height (et *line-height*) de l'élément de la liste LI et de l'ancre A est fixé.

```
.tags li, .tags a{
        float:left;
        height:24px;
        line-height:24px;
        position:relative;
        font-size:11px;
}
```

Ensuite nous allons styler l'élément ancre (a). Nous ajoutons des marges (margin), du padding et un coin arrondi sur le côté droit.

```
.tags a{
        margin-left:20px;
        padding:0 10px 0 12px;
        background:#0089e0;
        color:#fff;
        text-decoration:none;
        -moz-border-radius-bottomright:4px;
        -webkit-border-bottom-right-radius:4px;
        border-bottom-right-radius:4px;
        -moz-border-radius-topright:4px;
        -webkit-border-top-right-radius:4px;
        border-top-right-radius:4px;
}
```

Pour terminer la pointe du bord, nous ajoutons un pseudoélément *:before*. L'élément a une hauteur et une largeur définies à zéro, de cette façon nous utiliserons uniquement ses bordures. Pour "dessiner" une flèche pointant vers la gauche, nous allons afficher uniquement la bordure de droite.

```
.tags a:before{
        content:"";
        float:left;
        position:absolute;
        top:0;
        left:-12px;
        width:0;
        height:0;
```

```
        border-color:transparent #0089e0
transparent transparent;
        border-style:solid;
        border-width:12px 12px 12px 0;
}
```

Le dernier élément à ajouter est le pseudoélément *:after*. Cela fera office d'un trou rond. Ce que nous allons faire ici, c'est créer un carré vide, dont nous allons arrondir les angles. Enfin nous allons créer un cercle (et bien sûr nous le positionnerons avec *position:absolute*).

```
.tags a:after{
        content:"";
        position:absolute;
        top:10px;
        left:0;
        float:left;
        width:4px;
        height:4px;
        -moz-border-radius:2px;
        -webkit-border-radius:2px;
        border-radius:2px;
        background:#fff;
        -moz-box-shadow:-1px -1px 2px #004977;
        -webkit-box-shadow:-1px -1px 2px #004977;
        box-shadow:-1px -1px 2px #004977;
}
```

Nous allons aussi ajouter un état *:hover* pour l'ancre :

```
.tags a:hover{
        background:#555;
}
.tags a:hover:before{
        border-color:transparent #555 transparent
transparent;
}
```

Et voilà ! Amusez vous !

Retrouvez l'article d'Alen Grakalic traduit par Jérôme Debray en ligne : Lien 39

Les transformations en CSS3

Dans cet article, je vais présenter les transformations 2D utilisables via CSS3 et qui permettent un grand nombre de possibilités visuelles grâce à la propriété transform.
Quand ce sera nécessaire, il sera indiqué les préfixes des navigateurs pour les propriétés CSS.
Compatibilité : Chrome, Safari, Opera, Firefox.

1. Un point d'origine

Les différents exemples qui seront présentés peuvent avoir un point d'origine différent si on le spécifie, grâce à la propriété **transform-origin.** Cette propriété prend deux types de valeurs soit **numérique** (100 px, 50 %...) **soit alphabétique** (left, top, right, bottom).

Voici un exemple :

```
-webkit-transform-origin: 0 0;
-moz-transform-origin: 0 0;
-o-transform-origin: 0 0;
transform-origin: 0 0;
```

ou

```
-webkit-transform-origin: top left;
-moz-transform-origin: top left;
-o-transform-origin: top left;
transform-origin: top left;
```

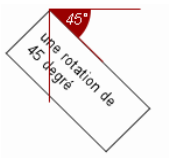
Par défaut, le point d'origine est *top left*.

2. A savoir

La propriété **transform** autorise plusieurs valeurs de transformation à la suite. Ainsi vous pourrez coupler les différents types de transformation entre eux :

```
-webkit-transform: skew(30deg, 15deg)
translate(10px, 0px) rotate(-30deg);
-moz-transform: skew(30deg, 15deg)
translate(10px, 0px) rotate(-30deg);
-o-transform: skew(30deg, 15deg) translate(10px,
0px) rotate(-30deg);
transform: skew(30deg, 15deg) translate(10px,
0px) rotate(-30deg);
```

3. La rotation



La rotation est possible via la propriété **transform** prenant pour valeur **rotate(x)**. L'argument de **rotate** correspond à la valeur de l'angle de rotation et peut être négatif.

```
-webkit-transform: rotate(45deg);
-moz-transform: rotate(45deg);
-o-transform: rotate(45deg);
transform: rotate(45deg);
```

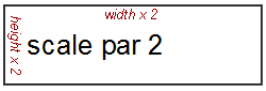
Attention : Sous les navigateurs Webkit (Chrome et Safari), on ne peut pas appliquer la transformation à n'importe quel tag HTML (par exemple le span ne permet pas de transformation de type rotation).

4. Le redimensionnement

La valeur **scale** de la propriété **transform** permet de dilater/redimensionner un élément. Cette valeur peut prendre **un ou deux arguments** : **scale(x)** ou **scale(x, y).**

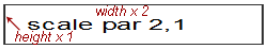
Avec une seule valeur, le **x** correspond au coefficient de dilatation en **largeur (width)** et en **hauteur (height).** Avec deux valeurs, **x** correspond à la **largeur (width)** et **y** à la **hauteur (height).**

```
-webkit-transform: scale(2);
-moz-transform: scale(2);
-o-transform: scale(2);
transform: scale(2);
```



ou

```
-webkit-transform: scale(2, 1);
-moz-transform: scale(2, 1);
-o-transform: scale(2, 1);
transform: scale(2, 1);
```



Il est à noter que l'on peut utiliser les sous-valeurs de **scale** : **scaleX** et **scaleY.**

Exemple:

```
transform: scaleX(2) scaleY(1);
```

ou

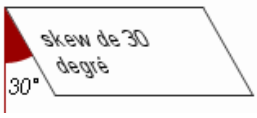
```
transform: scaleX(2);
```

5. La distorsion

La valeur **skew** de la propriété **transform** permet de tordre un élément. Cette valeur peut prendre **un ou deux arguments** (unité en degré).

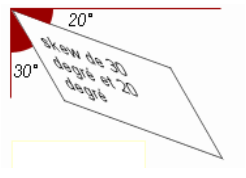
S'il n'y a qu'un argument, cela équivaut à une distorsion horizontale. S'il y a le deuxième argument, celui-ci contrôlera la distorsion verticale.

glissement horizontal :



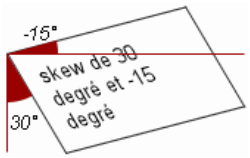

```
-webkit-transform: skew(30deg);
-moz-transform: skew(30deg);
-o-transform: skew(30deg);
transform: skew(30deg);
```

glissement horizontal et vertical :




```
-webkit-transform: skew(30deg, 20deg);
-moz-transform: skew(30deg, 20deg);
-o-transform: skew(30deg, 20deg);
transform: skew(30deg, 20deg);
```

Les arguments de la valeur **skew** peuvent être négatifs :




```
-webkit-transform: skew(30deg, -15deg);
-moz-transform: skew(30deg, -15deg);
-o-transform: skew(30deg, -15deg);
transform: skew(30deg, -15deg);
```

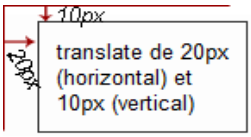
Tout comme **scale, skew** peut être utilisé sous cette forme :

```
transform:skewY(30deg) skewX(deg);
```

6. La translation

La valeur **translate** de la propriété **transform** permet de déplacer un élément en **x** et **y** par rapport à sa position d'origine. Cette valeur peut prendre un ou deux arguments.

Le premier argument correspond à une translation en **x** (horizontale) et le deuxième argument à une translation en **y** (verticale).




```
-webkit-transform: translate(20px, 10px);
-moz-transform: translate(20px, 10px);
-o-transform: translate(20px, 10px);
transform: translate(20px, 10px);
/*translation horizontale de 20px et translation
verticale de 10px*/
```

```
-webkit-transform: translate(20px);
-moz-transform: translate(20px);
-o-transform: translate(20px);
transform: translate(20px);
/*translation horizontale et verticale de 20px*/
```

Ces arguments peuvent être négatifs.

Tout comme **scale** et **skew, translate** peut être utilisé sous cette forme :

```
transform: translateY(20px) translateX(10px);
```

7. Exemples

Lien 40

Retrouvez l'article de Jérôme Debray en ligne : Lien 41

De la géométrie avec CSS

Grâce aux CSS et à la nouvelle norme CSS3, nous pouvons créer de plus en plus de formes telles que les carrés, les rectangles, les ronds, etc. Dans cet article, je vais présenter les différentes possibilités de formes faisables en CSS (du moins une liste non exhaustive).
Tous les exemples auront pour structure HTML cette base :

```
<div class="nom_de_la_forme">
</div>
```

Compatibilité : Chrome, Safari, Opera, Firefox 4, IE9.
Compatibilité partielle : Firefox 3.5 et IE8 (problème avec le border-radius).

1. Le carré

Ici, rien de bien compliqué, il suffit de définir une largeur et une hauteur identiques pour chaque côté de notre élément HTML.

```
.carre{
	width:200px;
	height:200px;
	background:#069;
}
```

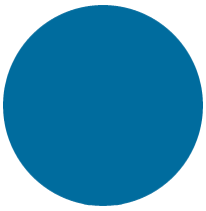
2. Le rectangle

Le principe est le même que le carré sauf que la largeur est plus grande que la hauteur (ou l'inverse selon l'effet désiré).

```
.rectangle{
	width:400px;
	height:200px;
	background:#069;
}
```

3. Le rond

Le rond est obtenu grâce à la propriété **border-radius** sur un élément carré. La valeur du **border-radius** sera égale à la moitié de la valeur d'un côté.

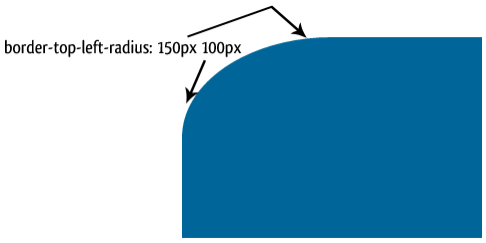


```
.rond{
	width:200px;
	height:200px;
	background:#069;
	-webkit-border-radius:100px;
	-moz-border-radius:100px;
	-o-border-radius:100px;
	border-radius:100px;
}
```

4. L'ovale

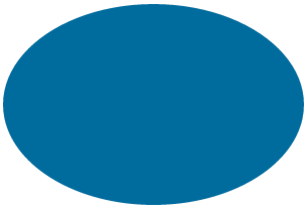
J'ai appris récemment que l'on pouvait donner deux valeurs au **border-radius** équivalant à l'arrondi de départ et l'arrondi d'arrivée d'un coin de bordure.

Par exemple :




```
border-top-left-radius : 150px 100px;
```

Le code ci-dessus va donner pour la bordure haut et droite, un arrondi commençant à 100px en radius sur le left pour arriver à 150px en radius sur le top.



Ainsi en appliquant ce procédé à toutes les bordures, on arrive à une forme ovale. Il faudra le faire sur un bloc ayant une forme rectangle.

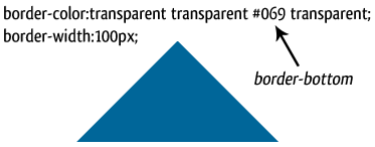
```
.ovale{
        width:300px;
        height:200px;
        background:#069;
        -webkit-border-radius:150px / 100px;
        -moz-border-radius:150px / 100px;
        -o-border-radius:150px / 100px;
        border-radius:150px / 100px;
}
```

équivaut à :

```
.ovale {
        width: 300px;
        height: 200px;
        background: #069;
        border-top-left-radius: 150px 100px;
        border-top-right-radius: 150px 100px;
        border-bottom-right-radius: 150px 100px;
        border-bottom-left-radius: 150px 100px;
}
```

5. Les triangles

Le principe du triangle est simple car il équivaut à manipuler la largeur des bordures ainsi que leur visibilité - par le biais de la couleur de la bordure - sur un élément HTML ayant des dimensions égales à 1px.




```
.triangle{
        width:1px;
        height:1px;
        border:1px solid #069;
        border-color:transparent transparent #069
transparent;
        border-width:100px;
}
```

Pour avoir un triangle orienté vers un autre côté, il suffit juste de changer les valeurs de **border-color**.

6. La transformation au service de la géométrie : le parallélogramme

Dans mon article précédent, je parlais des propriétés **transform** introduites en CSS3. Celles-ci nous permettent de modifier la forme de nos éléments (dans un plan 2D).

Voici un rappel des principales fonctions de transformation :

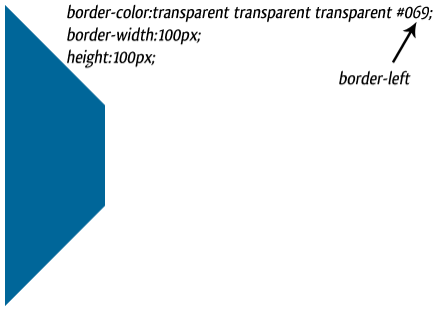
- skew ;
- rotate ;
- translate.

La fonction qui va nous intéresser sera la fonction **skew**. On part de la base d'un rectangle auquel on applique la transformation :


```
.parallelogramme{
        width:300px;
        height:100px;
        background:#069;
        -webkit-transform:skew(30deg);
        -moz-transform:skew(30deg);
        -o-transform:skew(30deg);
        tranform:skew(30deg);
}
```

7. Le trapèze

Un trapèze se fait sur la même base que le triangle mais au lieu d'avoir une hauteur ou une largeur à 1px, on a un des deux côtés à 1px et l'autre ayant une valeur supérieure à 1px (100px par exemple).




```
.trapeze{
        width:1px;
        height:100px;
        border:1px solid #069;
        border-color:transparent transparent
transparent #069;
        border-width:100px;
}
```

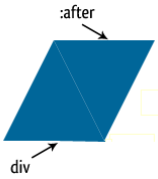
8. Les autres formes : la magie des pseudoéléments :after et :before

Jusque-là, on a vu la puissance du CSS/CSS3 pour produire de la géométrie. Grâce aux pseudoéléments **:after** et **:before** (ou **::after** et **::before**), on peut concevoir des formes beaucoup plus complexes.

Dans le tutoriel d'Alen Grakalic (tags pour vos billets blogs en css3 pure : Lien 39), on voit un peu ce que l'on peut obtenir grâce à ces pseudoéléments. Voyons ce que l'on peut en faire.

9. L'étoile

L'astuce est assez simple, car il suffit juste de créer un triangle. Puis grâce au pseudoélément **:after**, on refait un triangle mais dans le sens opposé.

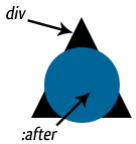


Une fois ces deux triangles créés, il suffit de positionner le triangle créé en **:after** afin de le superposer au deuxième triangle (via la propriété **position** de valeur **absolute**).

```
.etoile{
	background:#000;
	margin:100px auto;
	width: 0;
	height: 0;
	border-left: 50px solid transparent;
	border-right: 50px solid transparent;
	border-bottom: 100px solid #069;
	position: relative;
}
.etoile:after{
	content:'';
	border: 1px solid #069;
	border-left: 50px solid transparent;
	border-right: 50px solid transparent;
	border-top: 100px solid #069;
	width:0;
	height:0;
	position: absolute;
	top:30px;
	left:-50px;
}
```

10. Triangle et rond

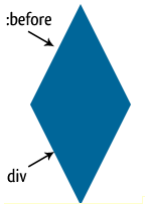
C'est le même principe que pour l'étoile.




```
.rondTriangle{
	margin:100px auto;
	width: 0;
	height: 0;
	border-left: 50px solid transparent;
	border-right: 50px solid transparent;
	border-bottom: 100px solid #000;
	position: relative;
}
.rondTriangle:after{
	content:'';
	background:#069;
	-webkit-border-radius:50px;
	-moz-border-radius:50px;
	-o-border-radius:50px;
	border-radius:50px;
	width:75px;
	height:75px;
	position: absolute;
	top:30px;
	left:-37px;
}
```

11. Le losange

Il s'agit de deux triangles juxtaposés.




```
.losange{
	margin:100px auto;
	width: 0;
	height: 0;
	border-left: 50px solid transparent;
	border-right: 50px solid transparent;
	border-bottom: 100px solid #069;
	position: relative;
}
.losange:before{
	content:'';
	border: 1px solid #069;
	border-left: 50px solid transparent;
	border-right: 50px solid transparent;
	border-top: 100px solid #069;
	width:0;
	height:0;
	position: absolute;
	top:100px;
	left:-50px;
}
```

On aurait pu faire ce losange en passant par un carré avec deux fonctions de transformation : **skew** et **rotate**.

12. Conclusion

A partir de là, on peut imaginer plein de possibilités de formes grâce aux nouvelles propriétés CSS3 et aux pseudoéléments.

Retrouvez l'article de Jérôme Debray en ligne : Lien 42

Créer un plugin de slideshow pour jQuery

Le but de ce tutoriel est de vous faire découvrir pas à pas comment construire votre premier plugin jQuery via un cas pratique : créer un slideshow.

1. Alors d'abord, un slideshow, c'est quoi ?

Je vous propose de commencer par la fin en faisant un détour par la démo, ici : Lien 43 (petite précision : vous le trouverez « peu » réactif aux clics, c'est normal, les photos ne sont pas hébergées sur mon serveur ce qui rend tout ça un peu plus lourd).

2. OK, c'est super mais on fait comment ?

On va faire simple, très simple. Tout d'abord, posons quelques règles de travail :

- je veux écrire le moins de HTML possible ;
- je veux que le nombre de slides soit dynamique ;
- je veux pouvoir positionner les liens où je le veux, aussi bien dans le slideshow qu'à l'extérieur ;
- je veux pouvoir personnaliser ces liens ;
- je veux que ça tourne tout seul, que l'utilisateur n'ait pas à cliquer.

Bon, on va s'arrêter là parce que ça ne fait déjà pas mal pour démarrer. Évidemment, on va faire en sorte qu'il soit suffisamment évolutif pour pouvoir revenir dessus. On va oublier pour cette première version la possibilité de faire un slide avec du texte, une page Web ou encore de la vidéo. On laisse tomber aussi l'idée qu'il soit dynamique, il faudra cliquer à la main pour faire défiler. Mais bon, je vous rassure. Ce n'est que provisoire.

On veut écrire le moins de HTML possible, OK, alors pour moi le plus simple, c'est ça :

index.htm

```
<html>
  <head>
    <title>Démo slideshow - Labs jQuery de
Mathieu ROBIN</title>
    <script type="text/javascript"
src="../jquery-1.4.3.min.js"></script>
    <script type="text/javascript"
src="jquery.slideshow.js"></script>
    <script type="text/javascript">
      <!-- Ici on mettra le code Javascript
nécessaire -->
    </script>
    <style type="text/css" media="screen">
      /* Et ici le code CSS nécessaire */
    </style>
  </head>
  <body>
    <div id="slideshow"></div>
  </body>
</html>
```

Vous comprendrez à la vue de ce code HTML plus que minimaliste que c'est bien votre code JS qui va se charger d'absolument tout. Bon évidemment il y aura du code CSS aussi pour rendre tout ça plus sympa. On va commencer par réaliser un joli cadre autour du slideshow et le positionner correctement, dans la partie CSS, on aura donc :

index.htm/CSS

```
div.slideshow{
        border:1px #000 solid;
        cursor:pointer;
        font-family:Arial;
        font-size:12px;
        height:420px;
        width:640px;
        position: absolute;
        top:50px;
        left:50px;
}
div.slideshow span.slideshow-link{
        background-color:#FFF;
        border:1px #000 solid;
        cursor:pointer;
        left:3px;
        line-height:22px;
        margin:3px;
        padding:2px 6px;
        position:relative;
        text-align:center;
        top:3px;
        vertical-align:top;
}
div.slideshow span.selected {
        font-weight: bold;
}
```

Je mets ce code CSS ici parce qu'il n'entre pas en compte dans l'exercice. Bien entendu, si vous êtes amené un jour à réaliser un « vrai » plugin jQuery avec des styles CSS spécifiques, pensez à les mettre dans un fichier. Maintenant on va appeler la fonction créatrice du slideshow, mais avant, il faut définir une structure de données JSON pour le contenu des slides. La suite va donc dans la partie script définie précédemment :

index.htm/Javascript

```
var data = {'imgs':[
        'http://farm1.static.flickr.com/79/244741
330_3f6a68924f_z.jpg',
        'http://farm1.static.flickr.com/177/42637
3909_15671ba709_z.jpg',
        'http://farm5.static.flickr.com/4096/4909
713714_736916b723_z.jpg',
        'http://farm5.static.flickr.com/4115/4767
```

```
003134_c75abfecb5_z.jpg'
]};
$(document).ready(function(){
        $('#slideshow').slideshow();
});
```

On commence simplement avec juste les adresses des images. Bien entendu, vous pourrez modifier le plugin pour charger du contenu Web, ajouter des liens, etc. On se contentera de ce petit bout de code, tout le reste est dans le plugin. Vous avez peut-être remarqué cette ligne-là dans le template HTML plus haut :

index.htm
```
<script type="text/javascript"
src="jquery.slideshow.js"></script>
```

Bon et bien dans le même dossier que votre fichier HTML, vous allez créer un fichier nommé « jquery.slideshow.js ». C'est lui qui contiendra le code du plugin.

3. On y arrive enfin !

Un plugin jQuery se construit toujours avec cette structure minimale là :

jquery.slideshow.js
```
(function($) {
  // Le code de votre plugin
})(jQuery);
```

Vous pouvez toujours essayer de la modifier, il y a effectivement des petites choses qu'on peut changer mais on sortirait trop du cadre de l'exercice et honnêtement : c'est à vos risques et périls de tenter le coup. On commence par créer une fonction qui englobera tout notre code et permettra de faire un premier test de notre plugin :

jquery.slideshow.js
```
(function($) {
  $.fn.slideshow = function(args) {
    alert('Ceci est mon premier plugin');
    return $(this);
  };
})(jQuery);
```

Si vous chargez la page HTML créée auparavant, vous devriez avoir une alerte de votre navigateur. Ne pas oublier la ligne de return qui sert à retourner l'objet jQuery lui-même. Comme ça vous pourrez admirer les jolis appels chaînés qui sont l'une des marques de fabrique de jQuery. Oui, je parle de ce genre de choses là :

Exemple d'appels chaînés
```
$('#maDiv').removeClass('classeAuPif').prev().
parent().next().addClass('classeAuPif');
```

On va découper le plugin en trois parties, une pour les variables, une pour les fonctions internes (je suis tenté de dire privé mais ce n'est pas exactement ça) et une troisième pour le corps principal du plugin. On va parler directement des variables. Je vous laisse regarder rapidement l'implémentation que je vous propose et on voit après le pourquoi.

jquery.slideshow.js
```
(function($) {
  $.fn.slideshow = function(args) {
  // Variables
    var defaults = {
      speed: 2000
    };
    var opts = $.extend(defaults, args);
    var imgsLength = data.imgs.length;
    var ele = this;
    var tokenValue = 0;
    // Fonctions
    // Main
    return $(this);
  };
})(jQuery);
```

J'utilise une variable intermédiaire (ele) pour désigner l'objet this parce que, dans le code du plugin, this ne désignera pas toujours la div visée. Je vous recommande de toujours utiliser cet intermédiaire. On a aussi tokenValue, cette variable servira à une chose très simple : contenir le numéro du slide en cours pour qu'on sache toujours où on en est. Nous aurons besoin à plusieurs reprises de connaître le nombre d'images à manipuler, pour faire plus simple et pour des raisons de performances, j'utilise la variable imgsLength.

4. T'as sauté args, defaults et opts

Disons que je les gardais pour la fin. Comme vous le savez peut-être, Javascript ne permet pas l'utilisation de valeurs par défaut. Mais jQuery propose une solution simple pour contourner ce problème. La fonction extend() fusionne deux tableaux en un. Vous pouvez l'utiliser de deux façons. Soit vous la mettez après un =, dans ce cas elle retourne le tableau créé (comme dans l'exemple), soit vous mettez l'appel seul, et dans ce cas, c'est le premier tableau passé en paramètre qui est modifié. Ici j'ai mis un seul élément dans le tableau, variable "speed", c'est l'intervalle de temps entre deux slides que j'ai mis à deux secondes. Vous pourrez donc faire cet appel à la place de l'ancien par exemple :

index.htm/Javascript
```
$(document).ready(function(){
  // On remplace $('#slideshow').slideshow(); par
  $('#slideshow').slideshow({speed: 100});
});
```

Mais bon vu que ce sont des millisecondes, ça risque d'être un peu rapide. Ceci dit, ça vous permettra d'aller dans l'autre sens aussi en mettant par exemple trois secondes de délai. **Avec cette façon de faire, vous pourrez donc passer un seul objet JSON qui contient tous vos paramètres et même les passer dans le désordre.** C'est beau n'est-ce pas.

Bon maintenant faut mettre une petite fonction qu'on va appeler de façon cyclique pour changer l'image. Hop, d'abord le code à insérer directement dans le code du plugin :

jquery.slideshow.js
```
var rotate = function() {
  $(ele).css('background-image','url(' +
data['imgs'][tokenValue] + ')');
  $('.selected').removeClass('selected');
  $('#slideshow-' +
tokenValue).addClass('selected');
  tokenValue++;
  if(imgsLength == tokenValue)
    tokenValue = 0;
  timer = setTimeout(this.rotate, opts.speed);
};
```

On a désormais une fonction qui s'appelle « rotate », que fait-elle? Elle retrouve l'élément visé par ele (anciennement this) et change la propriété CSS « background-image ». Si vous voulez afficher un script HTML, c'est cette instruction-là qu'il faudra remplacer. Au passage, j'en profite pour retirer la classe CSS « selected » de tout élément la possédant, pour ensuite l'appliquer à l'indicateur de position dans le slideshow. Si vous regardez la source CSS plus haut dans ce billet, vous verrez que la classe CSS « selected » ne fait que mettre en gras le texte. Vous pouvez aussi bien entendu modifier ça pour ajouter une couleur de fond par exemple. Vu qu'on souhaite se déplacer en cycle, il vaut mieux changer la valeur de tokenValue pour que l'image suivante (d'un point de vue du temps) ne soit pas la même. Au passage on s'assure que si on dépasse le nombre de photos, on revient à 0, pour éviter de ne rien afficher parce qu'on a oublié de s'arrêter.

4.1. Euh attends là, tu peux attaquer directement la variable data qui n'est pas dans le plugin ?

Oui. J'ai fait une petite erreur volontairement pour vous montrer qu'il est très important de faire attention à ce qu'on utilise quand on code un plugin jQuery. Ici data est une variable globale, je peux donc l'utiliser de partout, tout le temps. Imaginez ce que ça donnerait si on appelle slideshow() et plus loin dans le script de votre application vous changez la variable « data ». Hum, le joli bazar que ça va mettre. Exemple à ne pas suivre donc. Vous me passerez donc le tableau d'adresses d'images en paramètre de votre appel à slideshow. Je vous aide, l'appel :

index.htm/Javascript
```
$('#slideshow').slideshow(data);
```

Et la fonction corrigée :

jquery.slideshow.js
```
var imgsLength = opts.imgs.length;
var rotate = function() {
  $(ele).css('background-image','url(' +
opts['imgs'][tokenValue] + ')');
  $('.selected').removeClass('selected');
  $('#slideshow-' +
tokenValue).addClass('selected');
  tokenValue++;
  if(imgsLength == tokenValue)
    tokenValue = 0;
  setTimeout(rotate, opts.speed);
};
```

Voilà, là c'est propre. Et ça a moins de chances de se planter. J'ai oublié de vous parler de la dernière ligne. Je définis un timeOut (fonction JS de base, pas jQuery) qui exécute une fonction (premier paramètre) après un temps défini (second paramètre). Ici on appelle donc récursivement la même fonction « rotate », on a donc une boucle et comme délai, on utilise la valeur qui sera sortie de extend(), donc soit la valeur par défaut, soit la valeur possiblement passée en paramètre.

Passons au programme principal (partie Main, cf. plus haut). D'abord le code :

jquery.slideshow.js
```
$(ele).addClass('slideshow');
for(var i = 0; i < imgsLength; i++){
        $(ele).append('<span id="slideshow-' + i
+ '" class="slideshow-link">' + (i + 1) +
'</span>');
}
rotate();
```

Rien de bien sorcier ici. On ajoute la classe CSS « slideshow » à ele (voir partie CSS), on ajoute nos curseurs de position à notre slideshow. Et on fait le premier appel à la fonction rotate() définie précédemment.

5. Je voulais aussi que mon utilisateur puisse cliquer sur les curseurs pour aller à un slide bien précis

On y vient. On rajoute ce dernier bout de code à la partie « main » du plugin :

jquery.slideshow.js
```
$("div.slideshow").delegate("span.slideshow-
link", "click", function(pEvent){
        tokenValue = parseInt($
(pEvent.target).text()) - 1;
        $('.selected').removeClass('selected');
        $(pEvent.target).addClass('selected');
        $(ele).css('background-image','url(' +
opts['imgs'][tokenValue] + ')');
});
```

Il faut savoir que jQuery vous permet d'ajouter des handlers d'évènements sur des objets créés à la volée, tels que nos curseurs. Pour ça on utilise la fonction delegate() (Lien 44). On commence donc par récupérer la valeur du curseur cliqué, on change de position la classe CSS « selected » comme déjà vu dans rotate() et on change l'image utilisée en appelant le bon tableau (n'oubliez pas qu'on ne touche plus à data).

6. Voilà, votre plugin est terminé et fonctionne.

Parce que c'est sûrement un peu brouillon, voilà le code complet :

jquery.slideshow.js
```
(function($) {
  $.fn.slideshow = function(args) {
    // Variables
    var defaults = {
      speed: 2000,
      imgs:[]
    };
    var opts = $.extend(defaults, args);
    var imgsLength = opts.imgs.length;
    var ele = this;
    var tokenValue = 0;
    // Fonctions
```

```
    var rotate = function() {
      $(ele).css('background-image','url(' +
opts['imgs'][tokenValue] + ')');
      $('.selected').removeClass('selected');
      $('#slideshow-' + tokenValue).addClass
('selected');
      tokenValue++;
      if(imgsLength == tokenValue)
        tokenValue = 0;
      setTimeout(rotate, opts.speed);
    };
    // Main
    $(ele).addClass('slideshow');
    for(var i = 0; i < imgsLength; i++){
      $(ele).append('<span id="slideshow-' + i +
'" class="slideshow-link">' + (i + 1) +
'</span>');
    }
    rotate();
    $("div.slideshow").delegate("span.slideshow-
link", "click", function(pEvent){
      tokenValue = parseInt($
(pEvent.target).text()) - 1;
      $('.selected').removeClass('selected');
      $(pEvent.target).addClass('selected');
      $(ele).css('background-image','url(' +
opts['imgs'][tokenValue] + ')');
    });
    return $(this);
  };
})(jQuery);
```

7. Je peux télécharger les sources et les utiliser sur mon site ?

Bien sûr, même si c'est pour une utilisation commerciale. Les contenus de mes labs sont sous licence Creative Commons et je ne demande qu'une chose: qu'on respecte la paternité de ma création. Pour ceux qui ont des doutes sur la façon dont ils ont recopié mon code, ils peuvent fouiller les sources des pages, ils devraient tout trouver assez facilement.

7.1. Et les photos de la démo ?

J'ai utilisé des photos provenant du site FlickR (Lien 45), ces photos sont sous licence CreativeCommons, vous pouvez donc les utiliser comme bon vous semble tant que vous n'oubliez pas de respecter certaines règles. D'ailleurs, c'est mon tour de respecter la licence :

- « Blading in Paris » par David Dennis (Paternité, Partage Sous Condition initiale) : Lien 46
- « Ski jump 242 » par Shay Haas(Paternité, Pas d'utilisation commerciale, Partage Sous Condition initiale) : Lien 47
- « Surf during sunset » par Montse PB (Paternité) : Lien 48
- « Skydiving 2010, flying free » par divemasterking2000 (Paternité) : Lien 49

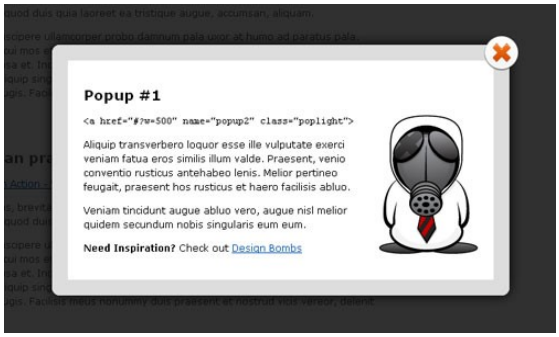
Retrouvez l'article de Mathieu Robin en ligne : Lien 50

Créez une fenêtre modale avec CSS et jQuery

Cet article est la traduction de l'article : Inline Modal Window w/ CSS and jQuery (Lien 51). Retrouvez toutes les traductions de Soh Tanaka disponibles sur http://sohtanaka.developpez.com/ (Lien 52).
Tout au long de cet article, l'auteur vous présente une méthode permettant de créer une popup CSS du même style que celles utilisées par la bibliothèque Lightview.

1. Présentation

Il existe de nombreux scripts de fenêtres modales (de type pop-up) simples à implémenter et élégantes. Mais la plupart du temps, ces scripts peuvent rentrer en conflit avec la logique propre de la page.
J'ai été récemment confronté à un cas où il m'était impossible d'utiliser les scripts comme fancybox (Lien 53) ou prettyPhoto (Lien 54). J'ai donc dû développer ma propre fenêtre modale pour y insérer du code **(X)HTML**.
Je vais vous expliquer comment j'ai procédé.



Voir une démo en ligne : Lien 55

2. La structure HTML

Commençons par ajouter une balise <a> avec les attributs suivants :

- href - #?w=500 : spécifie la largeur de la fenêtre ;
- rel : définit la relation avec la pop-up à ouvrir ;
- class="poplight" : classe **CSS** pour gérer les pop-up.

```
<a href="#?w=500" rel="popup_name"
class="poplight">En savoir plus</a>
```

Ensuite, nous ajoutons le code **(X)HTML** des fenêtres.
Vous pouvez les placer où vous voulez dans la page, pour ma part, j'ai opté pour les mettre en fin de code.
Notez bien que l'attribut id correspond à l'attribut rel de la balise <a>. Cela permet d'établir la relation entre le lien et la fenêtre correspondante.

```
<div id="popup_name" class="popup_block">
  <h2>Developpez.com</h2>
  <p>Soh Tanaka est traduit sur
developpez.com.</p>
</div>
```

3. La mise en forme avec le CSS

Le code **CSS** est commenté afin de vous expliquer comment il fonctionne.
Notez que nous ne précisons pas le margin pour la classe .popup_block : comme la taille de la fenêtre peut varier, c'est *jQuery* qui le calculera à l'étape suivante.

```
#fade { /*--Masque opaque noir de fond--*/
        display: none; /*--masqué par défaut--*/
        background: #000;
        position: fixed; left: 0; top: 0;
        width: 100%; height: 100%;
        opacity: .80;
        z-index: 9999;
}
.popup_block{
        display: none; /*--masqué par défaut--*/
        background: #fff;
        padding: 20px;
        border: 20px solid #ddd;
        float: left;
        font-size: 1.2em;
        position: fixed;
        top: 50%; left: 50%;
        z-index: 99999;
        /*--Les différentes définitions de Box
Shadow en CSS3--*/
        -webkit-box-shadow: 0px 0px 20px #000;
        -moz-box-shadow: 0px 0px 20px #000;
        box-shadow: 0px 0px 20px #000;
        /*--Coins arrondis en CSS3--*/
        -webkit-border-radius: 10px;
        -moz-border-radius: 10px;
        border-radius: 10px;
}
img.btn_close {
        float: right;
        margin: -55px -55px 0 0;
}
/*--Gérer la position fixed pour IE6--*/
*html #fade {
position: absolute;
}
*html .popup_block {
position: absolute;
}
```

4. Mise en place de jQuery

Pour ceux qui ne sont pas familiers avec jQuery (Lien 56), je vous invite à mieux connaitre cette librairie sur leur site pour comprendre comment elle fonctionne.

Vous pouvez choisir de télécharger (Lien 57) ou de le charger depuis le site Google.

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/
1.4.1/jquery.min.js"></script>
```

Après avoir chargé *jQuery*, vous pouvez ouvrir une nouvelle balise <script> et commencer votre code avec l'événement $(document).ready, ce qui permet au code *jQuery* d'être exécuté dès que le **DOM** est disponible.
Le reste du code nécessaire au script s'y trouvera.

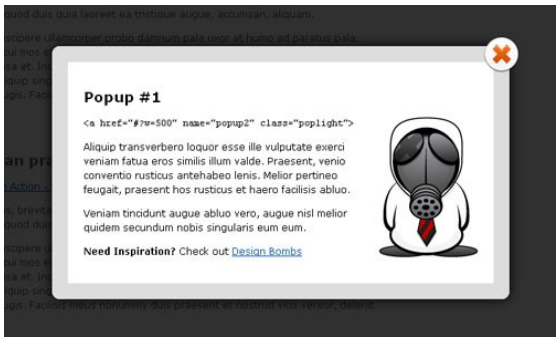
```
$(document).ready(function() {
  //Le code ici
});
```

5. La touche finale : le code jQuery

Le code suivant est commenté pour vous permettre de comprendre le fonctionnement du script.

```
//Lorsque vous cliquez sur un lien de la classe
poplight et que le href commence par #
$('a.poplight[href^=#]').click(function() {
  var popID = $(this).attr('rel'); //Trouver la
pop-up correspondante
  var popURL = $(this).attr('href'); //Retrouver
la largeur dans le href

  //Récupérer les variables depuis le lien
  var query= popURL.split('?');
  var dim= query[1].split('&');
  var popWidth = dim[0].split('=')[1]; //La
première valeur du lien

  //Faire apparaitre la pop-up et ajouter le
bouton de fermeture
  $('#' + popID).fadeIn().css({
    'width': Number(popWidth)
  })
  .prepend('');

  //Récupération du margin, qui permettra de
centrer la fenêtre - on ajuste de 80px en
conformité avec le CSS
  var popMargTop = ($('#' + popID).height() + 80)
/ 2;
  var popMargLeft = ($('#' + popID).width() + 80)
/ 2;

  //On affecte le margin
  $('#' + popID).css({
    'margin-top' : -popMargTop,
    'margin-left' : -popMargLeft
  });

  //Effet fade-in du fond opaque
  $('body').append(''); //Ajout du fond opaque
noir
  //Apparition du fond - .css({'filter' :
'alpha(opacity=80)'}) pour corriger les bogues de
IE
  $('#fade').css({'filter' :
'alpha(opacity=80)'}).fadeIn();

  return false;
});

//Fermeture de la pop-up et du fond
$('a.close, #fade').live('click', function()
{ //Au clic sur le bouton ou sur le calque...
  $('#fade , .popup_block').fadeOut(function() {
    $('#fade, a.close').remove();  //...ils
disparaissent ensemble
  });
  return false;
});
```

6. Démo et conclusion



Voir une démo en ligne : Lien 55

Si vous êtes à l'aise avec **jQuery** et que vous voyez des améliorations à apporter au code, n'hésitez pas à les proposer.
A l'inverse, si vous êtes débutant et que des points ne vous semblent pas clairs, faites la demande sur le forum.

Retrouvez l'article de Soh Tanaka traduit par Didier Mouronval en ligne : Lien 58

XPath 1.0: fonctionnement des prédicats

Cet article étudie le fonctionnement des prédicats qui permettent de poser des conditions dans le langage XPath. Il fait suite à XPath : Types, axes et éléments (Lien 59).

1. Généralités

Un prédicat commence par [et se termine par].

Un prédicat peut contenir des XPath et ainsi d'autres prédicats.

Un prédicat est une condition, on peut le comparer à la clause **WHERE** en SQL, il en diffère néanmoins par le fait qu'il ne porte pas sur l'intégralité du Xpath (sauf usage particulier de parenthèses) mais sur la combinaison axe+test qu'il suit directement : son contexte.

2. Opérateurs

2.1. Les opérateurs booléens

Les booléens sont true() et false(). Le noeud vide, la chaîne vide et zéro sont convertis en false().

- NON : c'est une fonction en XPath, **not(...)**, elle englobe la partie sur laquelle porte la négation ;
- OU : **or** ;
- ET : **and** ;
- EGAL et DIFFERENT : = et **!=** attention **!=** n'est pas la négation de =, comme cela sera détaillé plus tard.
- COMPARATEURS D'ORDRE : <=, <, >=, >.

2.2. Les opérateurs numériques

Si une de ces opérations est effectuée sur une chaîne de caractères la valeur renvoyée est **NaN** (Not a Number).

- ADDITION : + ;
- SOUSTRACTION : **-**, attention sur l'opérateur de soustraction, il faut toujours le faire précéder et suivre d'un espace sinon l'expression peut être confondue avec un nom d'élément ;
- MULTIPLICATION : * ;
- DIVISION: **div** ;
- MODULO: **mod**.

3. Test

Lors de test entraînant une comparaison tout noeud est converti en sa valeur textuelle. Celle-ci pourra être considérée comme un nombre (si sa forme le permet) ou une chaîne de caractères.

3.1. Test de valeur

Il porte sur la valeur textuelle du noeud sélectionné. On ne peut comparer que des types simples.
On notera ici l'importance du self::node() et de son raccourci «.».

Xpath : **//AA[.=1]**
```
<ROOT>
  <AA>1</AA>
  <AA>2</AA>
</ROOT>
```

Xpath : **//AA[. > 1]**
```
<ROOT>
  <AA>1</AA>
  <AA>2</AA>
</ROOT>
```

3.2. Test d'existence

On souhaite sélectionner un noeud en fonction de ses fils/père/attributs/etc. Dans ce cas, le XPath présent dans le prédicat sera évalué à partir du noeud sélectionné précédant ce même prédicat, **il ne sera donc pas précédé de** /.

Xpath : **//AA[DD] (tous les AA possédant au moins un fils DD)**
```
<ROOT>
  <AA>
    <BB>
      <CC/>
    </BB>
    <DD/>
  </AA>
  <AA>
    <BB/>
  </AA>
</ROOT>
```

Xpath : **//BB[*] (tous les BB possédant au moins un fils element)**
```
<ROOT>
  <AA>
    <BB>
      <CC/>
    </BB>
    <DD/>
  </AA>
  <AA>
    <BB/>
  </AA>
</ROOT>
```

Pour bien comprendre l'importance des chemins relatifs, le XPath suivant **//AA[/*//AA/DD]** se traduit par : *tous les AA à condition qu'il existe un AA descendant de la racine possédant au moins un fils DD.*

3.3. Test de position

On utilisera pour ceci une fonction XPath.

La fonction position() renvoie la position du noeud en lecture dans son contexte parent. La numérotation des positions en XPath commence à 1.

Xpath : **/ROOT/AA[position()=2]**
ou /ROOT/AA[2](écriture raccourcie) sélectionne le deuxième fils AA de ROOT

```
<ROOT>
  <AA>
    <BB>
      <CC/>
    </BB>
    <DD/>
  </AA>
  <AA>
    <BB/>
  </AA>
</ROOT>
```

Quand on teste les positions, sauf si on utilise les parenthèses comme indiqué dans le chapitre 4, le contexte parent est le noeud parent. Ainsi un XPath du type **//*[2]** ne renvoie pas le deuxième noeud de la sélection mais tous les noeuds sélectionnés précédemment qui sont des deuxièmes fils.

Xpath : **//*[2]**

```
<ROOT>
  <AA>
    <BB>
      <CC/>
    </BB>
    <DD/>
  </AA>
  <AA>
    <BB/>
  </AA>
</ROOT>
```

3.4. Différence entre = et !=

La différence entre = , != et leur négation se fait sensible lors de comparaison sur des ensembles.

Voyons déjà le comportement sur les ensembles.

Si on compare une valeur à un ensemble de noeuds via = , l'expression renvoie vrai si la valeur textuelle d'un des noeuds est égale à la valeur du test.

Xpath: **//a[.=//b]**

```
<r>
  <a>1</a>
  <a>5</a>
  <b>1</b>
  <b>2</b>
  <b>3</b>
</r>
```

Si on compare une valeur à un ensemble de noeuds via **!=** , l'expression renvoie vrai si la valeur textuelle d'un des noeuds est différente de la valeur du test.

Xpath: **//a[.!=//b]**

```
<r>
  <a>1</a>
  <a>5</a>
  <b>1</b>
  <b>2</b>
  <b>3</b>
</r>
```

Si on inclut l'expression avec = dans une négation, l'expression renvoie vrai si toutes les valeurs textuelles sont différentes de la valeur du test.

Xpath: **//a[not(.=//b)]**

```
<r>
  <a>1</a>
  <a>5</a>
  <b>1</b>
  <b>2</b>
  <b>3</b>
</r>
```

Si on inclut l'expression avec **!=** dans une négation , l'expression renvoie vrai si toutes les valeurs textuelles sont égales à la valeur du test.

Xpath: **//a[not(.!=//b)]**

```
<r>
  <a>1</a>
  <a>5</a>
  <b>1</b>
  <b>2</b>
  <b>3</b>
</r>
```

Xpath: **//a[not(.!=//b)]**

```
<r>
  <a>1</a>
  <a>5</a>
  <b>1</b>
  <b>1</b>
</r>
```

4. Parenthèses et contexte

Les parenthèses sont utilisables non seulement à l'intérieur des prédicats mais aussi sur les chemins Xpath.

A l'intérieur d'un prédicat les parenthèses suivent les règles de priorité classiques des opérations logiques.

La syntaxe XPath nécessite que la parenthèse gauche soit toujours placée en tête du chemin, ainsi :

(/AA/DD) est bon, **/AA(/DD)** ou **/AA/(DD)** est faux.

La parenthèse va permettre de considérer tout le Xpath englobé comme un ensemble de noeuds sans contexte.

Ainsi pour les prédicats, en particulier sur ceux effectuant des tests de position, on ne tiendra plus compte du contexte du noeud.

Quelques exemples avec et sans parenthèses pour comparaison :

Xpath : **/ROOT/AA/BB[position()=2]**

```
<ROOT>
  <AA>
    <BB/>
  </AA>
  <AA>
    <BB/>
    <BB/>
  </AA>
  <AA>
    <BB/>
  </AA>
</ROOT>
```

Xpath :(**/ROOT/AA/BB)[position()=2]**

```
<ROOT>
  <AA>
    <BB/>
  </AA>
  <AA>
    <BB/>
    <BB/>
  </AA>
  <AA>
    <BB/>
  </AA>
</ROOT>
```

Xpath : **/ROOT/AA/BB[position()=1]**

```
<ROOT>
  <AA>
    <BB/>
  </AA>
  <AA>
    <BB/>
    <BB/>
  </AA>
  <AA>
    <BB/>
  </AA>
</ROOT>
```

Xpath :(**/ROOT/AA/BB)[position()=1]**

```
<ROOT>
  <AA>
    <BB/>
  </AA>
  <AA>
    <BB/>
    <BB/>
  </AA>
  <AA>
    <BB/>
  </AA>
</ROOT>
```

5. Quelques exemples combinés

Après avoir vu en détail les différents types d'opérateurs et de tests, la démonstration sera complète avec quelques exemples de combinaisons de ces notions.

5.1. Un noeud avec au moins N fils

Si on souhaite un noeud avec N fils, cela signifie qu'il possède au moins un fils de position N.
On le traduira par une syntaxe du type : Chemin_XPath[*[position()=N]].

Xpath : **/ROOT/AA[*[position()=2]]**

```
<ROOT>
  <AA>
    <BB/>
  </AA>
  <AA>
    <BB/>
    <BB/>
    <BB/>
  </AA>
  <AA>
    <BB/>
    <BB/>
  </AA>
</ROOT>
```

5.2. Enième élément par ordre d'apparition

Il n'est pas toujours évident de repérer le énième élément d'une structure arborescente. Pour ceci les parenthèses nous seront d'une grande aide.

Xpath : **(//BB)[3]**

```
<ROOT>
  <BB>
    <BB/>
  </BB>
  <AA>
    <BB/>
    <BB/>
  </AA>
</ROOT>
```

5.3. Fils B d'un élément A si celui-ci possède aussi un fils C

Nous sommes simplement dans le cas où, au lieu d'être à la fin du XPath, le prédicat est au milieu.

Xpath : **/ROOT/AA[CC]/BB**

```
<ROOT>
  <AA>
    <BB/>
  </AA>
  <AA>
    <BB/>
    <CC/>
  </AA>
</ROOT>
```

5.4. Fils C d'un élément B ou A

Il existe différentes façons de coder ceci, la méthode présentée ici a l'avantage de rester simple et d'être très peu coûteuse.

Xpath : **/ROOT/*[self::AA or self::BB]/CC**

```
<ROOT>
  <BB>
    <CC/>
```

```
  </BB>
  <EE>
    <CC/>
  </EE>
  <AA>
    <BB/>
    <CC/>
  </AA>
</ROOT>
```

5.5. Suppression de doublons

La technique est simple bien que gourmande en ressources. Pour éliminer les doublons, il suffit d'éliminer les éléments qui ont la particularité d'être précédés par un élément qui leur est identique.
Nous utiliserons pour ceci les axes preceding ou preceding-sibling.

Xpath: **/r/*[not(preceding-sibling::*=.)]**

```
<r>
  <a>1</a>
  <a>5</a>
  <b>1</b>
  <b>2</b>
  <b>5</b>
</r>
```

Retrouvez l'article d'Erwan Amoureux en ligne : Lien 60

XPath 1.0: liste des fonctions

Cet article liste et décrit les fonctions et leur mécanisme en Xpath 1.0.
Quelques exemples de combinaisons de fonctions y ont été adjoints.

1. Paramètres de fonctions

Les fonctions XPath prennent en paramètre aussi bien des types simples que des nodesets. Quand le type du paramètre passé n'est pas celui demandé un transtypage implicite est effectué :

Paramètre demandé de type string

- le paramètre passé est un numérique : on applique la fonction string sur ce numérique ;
- le paramètre passé est un booléen : on applique la fonction string sur ce booléen, on renvoie la chaîne **false** ou **true** ;
- le paramètre passé est un nodeset vide : on renvoie une chaîne vide ;
- le paramètre passé est un nodeset d'un noeud : on applique la fonction string sur la valeur textuelle du noeud ;
- le paramètre passé est un nodeset de plusieurs noeuds : on applique la fonction string sur la valeur textuelle du premier noeud.

Paramètre demandé de type numérique

- le paramètre passé est une chaîne : on applique la fonction number, si la chaîne est vide ou contient tout autre caractère qu'un nombre (espace compris) la fonction renvoie la chaîne **NaN** ;
- le paramètre passé est un booléen : on applique la fonction number sur ce booléen, si le booléen vaut vrai (**true**), on renvoie 1, faux (**false**) on renvoie 0 ;
- le paramètre passé est un nodeset vide : on renvoie **NaN** ;
- le paramètre passé est un nodeset d'un noeud : on applique la fonction string sur la valeur textuelle du noeud, puis on applique la fonction number sur le résultat ;
- le paramètre passé est un nodeset de plusieurs noeuds : on applique la fonction string sur la valeur textuelle du premier noeud puis on applique la fonction number sur le résultat.

Paramètre demandé de type booléen

- le paramètre passé est une chaîne : on applique la fonction boolean, si c'est une chaine vide on renvoie **false**, **true** autrement ;
- le paramètre passé est un numérique : on applique la fonction boolean, s'il vaut zéro on renvoie **false**, **true** autrement ;
- le paramètre passé est un nodeset vide : on applique la fonction boolean et on renvoie **false** ;
- le paramètre passé est un nodeset d'un ou plusieurs noeuds : on applique la fonction boolean et on renvoie **true**.

Pour les paramètres de fonction de type **nodeset**, il n'y a pas de transtypage. Le comportement varie suivant la fonction. Certaines fonctions prennent le noeud en lecture par défaut si aucun noeud n'est fourni en paramètre. Attention sur ce point, un nodeset vide (chemin XPath ne ramenant pas de noeud) **n'est pas l'équivalent** d'une absence de passage paramètre.

2. Fonctions de conversions

2.1. String

string(*object* ?): *retourne* ***string***

Elle convertit un objet en chaîne de caractères selon les règles suivantes.

Un ensemble de noeuds est converti en chaîne de caractères en retournant la valeur textuelle du premier noeud de l'ensemble passé en argument . Si l'ensemble de noeuds est vide, une chaîne vide est retournée.

Cette fonction convertit aussi les nombres (entiers, décimaux) et les booléens (false, true).

Si un ensemble de noeuds est passé en paramètre à une fonction où une chaîne est demandée, cette fonction est appliquée par défaut pour la conversion.

2.2. Number

number(*object* ?): *retourne* ***nombre***

Elle convertit l'argument passé en nombre.

Les booléens "vrai" (true) sont convertis en 1 ; les booléens "faux" (false) sont convertis en 0.

Un ensemble de noeuds est d'abord converti en chaîne de caractères comme avec la fonction **string** et cette chaîne est ensuite considérée comme argument à la fonction **number.**

Si l'argument est omis, la valeur retournée par défaut est celle qui aurait été obtenue en considérant un ensemble de noeuds contenant le noeud en lecture.

2.3. Boolean

boolean(*object*): *retourne* ***boolean***

Elle convertit l'argument passé en booléen.

Un nombre est vrai (true) si et seulement s'il n'est ni un zéro positif ou négatif, ni un **NaN**.

Un ensemble de noeuds est vrai (true) si et seulement s'il n'est pas vide.

Une chaîne de caractères est vrai (true) si et seulement si sa longueur n'est pas nulle.

Un objet d'un type autre que les quatre de base est converti selon des règles spécifiques à chaque type.

Tout test Xpath utilise la conversion **boolean(..)** par défaut.

3. Fonctions d'ensembles de noeuds

3.1. Position

position() : retourne ***nombre***

Elle retourne un nombre égal à la position du noeud en lecture par rapport à son contexte d'évaluation. Celui-ci étant généralement son parent sauf en cas de chemin XPath parenthésé : Lien 61.

Pour les détails voir l'article précédent : 3-C Test de position (Lien 62)

3.2. Last

last() : retourne ***nombre***

Elle retourne la taille du contexte d'évaluation de l'expression, soit le nombre de noeuds contenus dans la dernière partie du Xpath à laquelle elle succède ou dans la parenthèse : Lien 61.

Récupérer le dernier fils d'un noeud :

Xpath : **/ROOT/AA/BB[position()=last()]**

```
<ROOT>
  <AA>
    <BB/>
  </AA>
  <AA>
    <BB/>
    <BB/>
    <BB/>
  </AA>
  <AA>
    <BB/>
    <BB/>
  </AA>
</ROOT>
```

Récupérer les noeuds qui ont moins de N fils :

Xpath : **/ROOT/AA[BB[last()<2]]**

```
<ROOT>
  <AA>
    <BB/>
  </AA>
  <AA>
    <BB/>
    <BB/>
    <BB/>
  </AA>
  <AA>
    <BB/>
    <BB/>
  </AA>
</ROOT>
```

3.3. Count

count(*node-set*) : *retourne* ***nombre***

Elle retourne le nombre de noeuds de l'ensemble passé en argument.

Soit le XML :

```
<?xml version="1.0" encoding="UTF-8"?>
<ROOT>
        <AA/>
        <BB/>
        <AA/>
        <AA/>
        <BB/>
</ROOT>
```

count(/ROOT/AA) renvoie 2.

count(/ROOT/BB) renvoie 3.

count(/ROOT/*) renvoie 5.

3.4. Name

name(*node-set* ?) : *retourne* ***string***

Elle retourne une chaîne contenant un nom qualifié (nom complet y compris le préfixe) représentant le nom du premier noeud de l'ensemble passé en argument. Si aucun noeud n'est passé en argument, le noeud utilisé par défaut

sera le noeud en lecture soit l'équivalent de **self::node()**.

Nous avons vu dans le cours sur le fonctionnement des prédicats comment on pouvait sélectionner des noeuds de même niveau mais de nom différent notamment grâce à l'axe **self::**
Cette technique ne fonctionne néanmoins que lorsque les noms des noeuds recherchés sont constants, ce qui, dans des contextes XSLT ou XQuery par exemple, n'est pas toujours le cas. Pour ceci on utilisera alors la fonction **name()**.

Xpath : **/ROOT/AA/*[name()='BB']**

```
<ROOT>
  <AA>
    <BB/>
    <CC/>
  </AA>
<ROOT>
```

3.5. Local-name

local-name(*node-set* ?) : *retourne* ***string***

Elle retourne la partie locale du nom expansé (le nom sans le préfixe) du premier noeud de l'ensemble passé en argument. Si aucun noeud n'est passé en argument, le noeud utilisé par défaut sera le noeud en lecture soit l'équivalent de **self::node()**.

Comparaison entre name() et local-name() :

Xpath : **/ROOT/*/*[name()="BB"]**

```
<ROOT test:xmlns="URI_de_test">
  <AA>
    <test:BB/>
  </AA>
  <test:AA>
    <BB/>
    <test:BB/>
    <BB/>
  </AA>
</ROOT>
```

Xpath : **/ROOT/*/*[local-name()="BB"]**

```
<ROOT test:xmlns="URI_de_test">
  <AA>
    <test:BB/>
  </AA>
  <test:AA>
    <BB/>
    <test:BB/>
    <BB/>
  </AA>
</ROOT>
```

3.6. Namespace_uri

namespace-uri(*node-set* ?) : *retourne* ***string***

Elle retourne l'URI du namespace du premier noeud du nodeset passé en paramètre. Si aucun noeud n'est passé en argument, le noeud utilisé par défaut sera le noeud en lecture soit l'équivalent de **self::node()**.

Xpath : **//*[namespace-uri()="URI_de_test"]**

```
<ROOT test:xmlns="URI_de_test">
  <AA>
    <test:BB/>
  </AA>
  <test:AA>
    <BB/>
    <test:BB/>
    <BB/>
  </AA>
</ROOT>
```

4. Fonctions de chaînes de caractères

4.1. Concat

concat(*string* , *string* , *string* *) : *retourne* ***string***

Elle retourne le résultat de la concaténation des arguments. Attention, lui passer un nodeset de plusieurs noeuds en paramètre ne permettra pas la concaténation de leur valeur textuelle. Chaque nodeset est considéré comme un seul argument et à ce titre seul le premier noeud est traité.

Exemple :

concat("AB","CDE","EF") retourne ABCDEF

4.2. Starts-with

starts-with(*string* , *string*) : *retourne* ***boolean***

Elle retourne la valeur booléenne true si la première chaîne de caractères passée en argument commence par la chaîne de caractères passée en deuxième argument.

Exemple :

starts-with("ABCDE","ABC">) retourne true.

starts-with("ABCDE","B") retourne false.

4.3. Contains

contains(*string* , *string*) : *retourne* ***boolean***

Elle retourne true si la première chaîne de caractères passée en argument contient la chaîne de caractères passée en deuxième argument sinon retourne la valeur false.

Exemple :

contains("ABCDE", "BC") retourne true.

contains("ABCDE", "Z") retourne false.

4.4. Substring-before

substring-before(*string* , *string*) : *retourne* ***string***

Elle retourne la sous-chaîne de caractères du premier argument qui précède la première occurrence de la deuxième chaîne de caractères dans le premier argument, ou une chaîne vide si le premier argument ne contient aucune occurrence de la deuxième.

Exemple :

substring-before("1999/04/01","/") retourne 1999.

4.5. Substring-after

substring-after(*string* , *string*) : *retourne* ***string***

Elle retourne la sous-chaîne de caractères du premier argument qui suit la première occurrence de la deuxième chaîne de caractères dans le premier argument, ou une chaîne vide si le premier argument ne contient aucune occurrence de la deuxième.

Exemple :

substring-after("1999/04/01","/") retourne 04/01, et **substring-after("1999/04/01","19")** retourne 99/04/01.

4.6. Substring

substring(*string* , *number* , *number* ?) : *retourne* ***string***

Elle retourne la sous-chaîne du premier argument commençant à la position spécifiée par le second argument et de longueur spécifiée par le troisième argument.

Exemple :

substring("12345",2,3) retourne 234.

Si le troisième argument n'est pas spécifié, la fonction retourne la sous-chaîne comprise entre la position de départ spécifiée et la fin de la chaîne de caractères initiale.

Exemple :

substring("12345",2) retourne 2345.

4.7. String-length

string-length(*string* ?) : *retourne* ***nombre***

Elle retourne le nombre de caractères de la chaîne. Si l'argument est omis, la valeur retournée est égale à la longueur de valeur textuelle du noeud en lecture.

4.8. Normalize-space

normalize-space(*string* ?) : *retourne* ***string***

Elle retourne la chaîne de caractères passée en argument après y avoir normalisé les espaces blancs : suppression des espaces en début et fin et remplacement des séquences de blancs successifs par un seul caractère blanc. Si l'argument est omis, la fonction retourne la chaîne obtenue en ayant utilisé comme argument la valeur textuelle du noeud en lecture.

normalize-space(' test avec nombreux espaces ') retourne la chaine : "*test avec nombreux espaces*".

4.9. Translate

translate(*string* , *string* , *string*) : *retourne* ***string***

Elle retourne la première chaîne de caractères passée en argument dans laquelle les occurrences des caractères de la deuxième chaîne sont remplacées par les caractères correspondant aux mêmes positions de la troisième chaîne.

Exemple :

translate("bar","abc","ABC") retourne la chaîne BAr.

Si l'un des caractères du deuxième argument n'a pas de position correspondante dans le troisième (parce que le deuxième argument est plus long que le troisième), alors les occurrences de ce caractère sont supprimées du premier argument.

Exemple :

translate("--aaa--","abc-","ABC") retourne AAA.

Si un caractère apparaît plus d'une fois dans la deuxième chaîne, alors c'est la première occurrence de ce caractère qui détermine la règle de transformation.

Exemple :

translate("bar","abbc","AcBC") retourne cAr.

Si la chaîne passée en troisième argument est plus longue que la deuxième, alors, les caractères excédentaires sont ignorés.

Exemple :

translate("bar","abc","ABCr") retourne BAr.

5. Fonctions numériques

5.1. Sum

sum(*node-set*) : *retourne* ***nombre***

La fonction **sum** retourne la somme, pour tous les noeuds de l'ensemble passé en argument, du résultat de la conversion en numérique de leur valeur textuelle. Si un ou plusieurs noeuds sélectionnés ne sont pas convertibles la fonction renverra **NaN**.

Attention de ne pas oublier que la conversion d'un noeud vide n'est pas zéro mais **NaN**.

Exemple :

```
<ROOT>
    <AA>
    1
    </AA>
    <AA>
    2
    </AA>
</ROOT>
```

sum(/ROOT/AA) retourne 3

5.2. Floor

floor(*number*) : *retourne* ***nombre***

La fonction **floor** retourne le plus grand nombre entier inférieur à l'argument du côté de l'infini positif. Soit l'entier immédiatement inférieur à sa gauche.

Exemple : **floor(4.2)** retourne 4 ; **floor(-4.2)** retourne -5.

5.3. Ceiling

ceiling(*number*) : *retourne* ***nombre***

La fonction ceiling retourne le plus petit (du côté de l'infini négatif) nombre entier qui ne soit pas inférieur à l'argument. Soit l'entier immédiatement supérieur à sa droite.

Exemple : **ceiling(4.2)** retourne 5 ; **ceiling(-4.2)** retourne -4.

5.4. Round

round(*number*) : *retourne* ***nombre***

retourne le nombre entier le plus proche de l'argument. Si deux nombres sont possibles, alors celui des deux qui est le plus proche de l'infini positif (à sa droite) est retourné.

Exemple : **round(4.2)** retourne 4 ; **round(4.6)** retourne 5 ; **round(4.5)** retourne 5 ;
round(-4.2) retourne -4 ; **round(-4.6)** retourne -5 ; **round(-4.5)** retourne -4.

6. Fonctions booléennes

6.1. Not

not(*boolean*) : *retourne* ***boolean***

La fonction **not** retourne l'inverse de la valeur du booléen passé en argument : vrai (true) si l'argument est faux et vice-versa.

6.2. True

true() : *retourne* ***boolean***

La fonction **true** retourne true.

Cette fonction est très peu utilisée, la conversion par transtypage étant la règle.

6.3. False

false() : *retourne* ***boolean***

La fonction **false** retourne false.

Cette fonction est très peu utilisée, la conversion par transtypage étant la règle.

7. Exemples de combinaisons

7.1. Recherche de balise dont le nom commence par...

Xpath : **//*[starts-with(name(),'B')]**

```
<ROOT >
   <AA>
      <BB/>
   </AA>
   <BA>
      <BB/>
      <CB/>
      <DB/>
   </BA>
</ROOT>
```

7.2. Recherche de contenu de chaîne dont on exclut certains caractères

Xpath : **/ROOT/*[contains(translate(.,'0123456789',''),'testA')]**

```
<ROOT>
   <A>test128A ;</A>
   <A>test1728B ;</A>
   <A>test28A ;</A>
   <A>essai196A ;</A>
   <A>test12C ;</A>
   <A>essai8A ;</A>
   <A>test12899A ;</A>
</ROOT>
```

7.3. Comparaison de dates

On va rester ici dans un cas simple, une date toujours formatée jj/mm/aaaa, mais il est tout à fait possible de traiter d'autres formats moins réguliers en s'appuyant sur le séparateur et des combinaisons de substring-before ou after.

Xpath : **/ROOT/A[concat(substring(., 7, 4),substring(., 4, 2),substring(., 1, 2))<20071201]**

```
<ROOT>
   <A>01/12/2006</A>
   <A>01/12/2007</A>
   <A>01/11/2007</A>
</ROOT>
```

7.4. Eliminer les noeuds non-numériques lors d'une somme

Xpath : **sum(/ROOT/A[number(.)!='NaN'])** retourne 37.

```
<?xml version="1.0" encoding="UTF-8"?>
<ROOT>
        <A>non renseigné</A>
        <A>absent</A>
        <A>10</A>
        <A>15</A>
        <A>12</A>
        <A>non renseigné</A>
        <A/>
</ROOT>
```

Retrouvez l'article d'Erwan Amoureux en ligne : Lien 63

Le comité ISO C++ valide le Draft final de la norme C++ 0X

Son nom sera C++ 2011

Les travaux pour la définition de la nouvelle norme pour le langage de programmation C++ sont enfin achevés et validés.

La norme, qui remplacera celle de 1997, et dont la publication initiale était prévue au plus tard pour 2010, vient de franchir un cap majeur. Le comité de normalisation ISO C++ vient en effet d'approuver les dernières modifications techniques lors d'une réunion qui s'est tenue du 21 au 25 mars à Madrid en Espagne, sur le Draft final (Final Commitee Draft) et sur un Draft international (Final Draft International Standard – FDIS).

Pour Herb Sutter, président du comité ISO C++, le FDIS est de «très bonne qualité », ce qui, en quelque sorte, pourrait justifier le retard accusé dans sa validation. « *Nous avons pris beaucoup plus de temps pour produire la seconde norme du C++. C'est en partie à cause de ses fonctionnalités ambitieuses, et surtout sa qualité [...] Cette norme est largement considérée comme le document FDIS de plus haute qualité que nous n'ayons jamais élaboré* » écrit-il sur son blog.

Au menu, des changements comme l'abandon des clauses new et explicit pour la gestion des overload, la rationalisation de l'utilisation de noexcept dans la bibliothèque ou la modification des règles de recherche de **Begin** et **end** pour un **range-for**.

On notera également la suppression de plusieurs spécifications jugées obsolètes.

La publication officielle de la norme est prévue pour cette année, si le FDIS est validé lors d'une ultime réunion à Genève.

Le nouveau standard aura finalement pour nom de code C++ 2011, mettant ainsi fin à toutes les spéculations, et à toutes les plaisanteries : Lien 64.

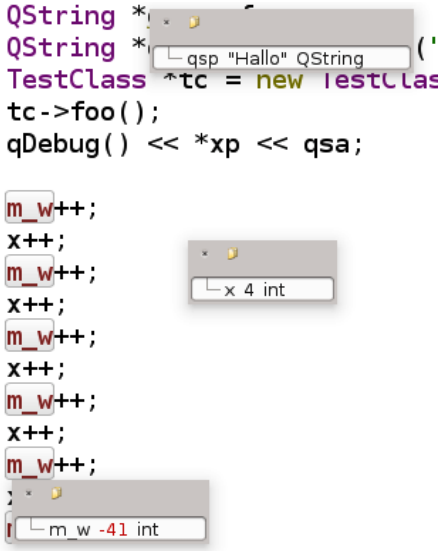
Source : Blog Herb Sutter : Lien 65

Commentez cette news de Romaric Hinault en ligne : Lien 66

Qt Creator 2.2 en beta

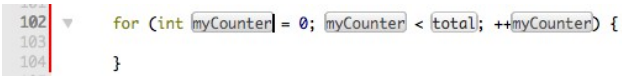
La dernière version de Qt Creator, la 2.2, arrive en beta aujourd'hui. En plus des habituelles corrections de bogues de la version précédente, elle apporte surtout quelques nouvelles fonctionnalités très utiles pour un EDI, rendant Qt Creator encore plus puissant et flexible.

Notons l'amélioration du support de QML (principalement dans l'édition textuelle de code QML), du débogage (le moteur CDB autorise maintenant le débogage d'applications 32 et 64 bits séparément, ainsi que le débogage de code mixte C++/Qt) et des outils de compilation. Aussi, les outils de débogage sont maintenant punaisables !



Bien d'autres choses sont encore prévues pour la version finale de Qt Creator 2.2, comme l'intégration d'outils externes améliorée, la configuration de chaînes de compilation manuelle, le changement des définitions de types MIME, ce qui rendra possible l'utilisation d'extensions différentes pour les fichiers source ou d'en-tête. Ce paramétrage se situe dans la page Environment > MIME Types des préférences.

Plus en détail, on trouve les snippets configurables, ajoutés aux différents éditeurs, qui fournissaient déjà moult snippets prédéfinis, que ce soit en C++ ou en QML. On peut les configurer dans les préférences, onglet Text Editor > Snippets.



Sources : Lien 67, Lien 68

Commentez cette news de Thibaut Cuvelier en ligne : Lien 69

La documentation de Qt 4.7 en français et avec arbre des classes

Nouvelle mise à jour sur toutes les pages de la documentation. Cette fois, seules les classes ont été touchées par la nouvelle fonctionnalité.

Depuis longtemps, plus aucun arbre des classes de Qt n'était disponible (la dernière version était pour Qt 4.3 (Lien 70), ça ne fait pas très récent). Voici la faille colmatée, avec un arbre interactif : chaque classe dispose d'un lien vers sa partie de l'arbre, cette partie montre ses parents et ses enfants directs. On peut afficher les parents et enfants de chacune de ces classes, pour autant qu'elle en ait. À chaque fois, un autre lien permet d'accéder directement à la documentation de la classe.

Bon amusement !

Un petit exemple : QboxLayout (Lien 71). Le lien pour y accéder se situe juste avant la table des matières de la page de la documentation : Lien 72.

Commentez cette news de Thibaut Cuvelier en ligne : Lien 73

PySide 1.0.0 en version finale

Le binding Python de Qt supporte aussi Qt Quick

Quelques jours après la sortie de Qt 4.7.2, le binding Python promu par Nokia, PySide, se met sur son trente-et-un et affiche la version finale de la 1.0.0, après un long cycle de développement, deux semaines après la release candidate, la communauté ayant apporté énormément au développement de cette version.

S'achève donc ainsi la période de correction des bogues, des régressions et des autres dysfonctionnements en tout genre : l'ajout de nouvelles fonctionnalités va pouvoir reprendre. Il est prévu notamment pour cette série d'ajouter le support de Python 3. Aussi, l'amélioration du code reprendra de plus belle, pour diminuer l'occupation mémoire et améliorer les performances.

D'ores et déjà, une série de paquets pour diverses plateformes sont disponibles : la plupart des distributions Linux répandues, Windows, OS X et Maemo 5 peuvent déjà passer à PySide 1.0.0 sans attendre.

La date de la prochaine version n'a pas été annoncée ; on pourrait s'attendre à voir des nouvelles d'ici à deux semaines, comme précédemment.

PySide 1.0.0 est disponible sur cette page : Lien 74
Source : Lien 75

Commentez cette news de Thibaut Cuvelier en ligne : Lien 76

Présentation de GLC_lib

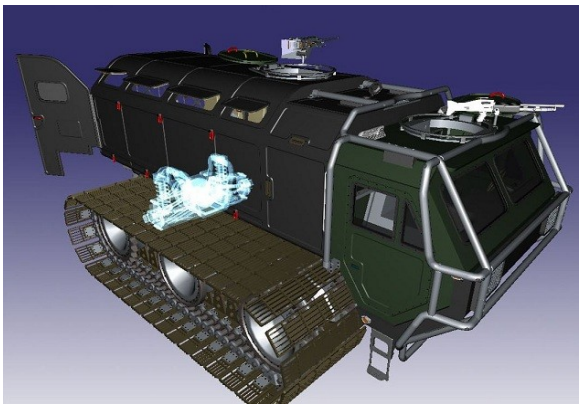
Ce tutoriel fournit une présentation de la bibliothèque GLC_lib, puis un exemple de programmation d'une application de visualisation 3D OpenGL utilisant GLC_lib.

1. Présentation de GLC_lib

1.1. Présentation générale

GLC_lib est une bibliothèque utilisant OpenGL et Qt4. Elle permet de créer et de visualiser en temps réel des scènes 3D. On peut donc dire que GLC_lib est un moteur 3D.

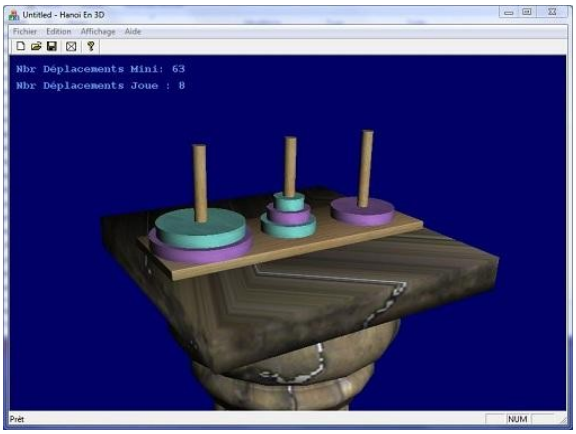
GLC_lib permet d'afficher en temps réel des scènes 3D complexes (100 000 pièces, 40 000 000 de triangles).



Exemple d'une scène contenant 241 pièces et 849 949 triangles

1.2. Bref historique

Le développement de GLC_lib a commencé en 2003 dans le but de créer des applications OpenGL. La première version de cette bibliothèque utilisait les MFC.



Les tours de Hanoï en 3D utilisant la première version de GLC_lib

En 2005, suite à la publication de Qt4 pour Windows sous licence GPL, le code de GLC_lib a été porté sous Qt4. Depuis la version 2.0, GLC_lib est sous licence LGPL.

1.3. Caractéristiques

GLC_lib permet la construction et le rendu d'une scène tridimensionnelle composée de centaines de milliers de maillages.

Pour le moment, GLC_lib n'est pas adaptée pour la représentation d'objets animés.

1.3.1. Visualisation d'un maillage

GLC_lib permet d'afficher des géométries composées de lignes, de triangles, de triangle strips et de triangle fans. Toutes ces géométries peuvent être affichées en utilisant les vertex array compatibles OpenGL 1.0 ou les Vertex Buffer Object à partir d'OpenGL 1.4.

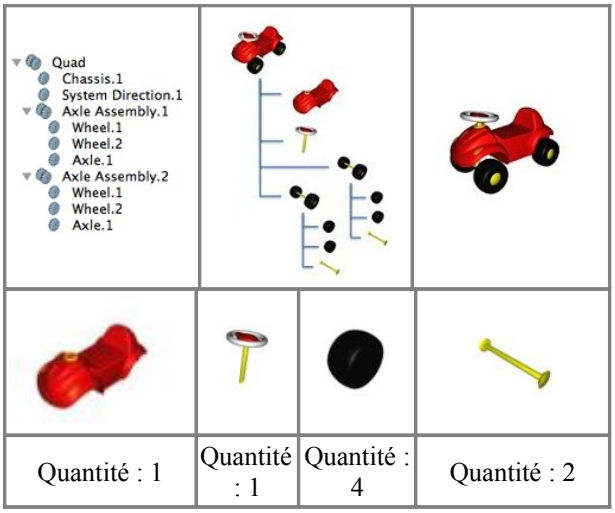
La structure de maillage de GLC_lib permet de représenter un objet solide ainsi que ses arêtes. Voici la liste de ses possibilités :

- affichage de polylignes ;
- affichage de triangles, triangle strips et triangle fans ;
- utilisation d'un nombre quelconque de matériaux opaques ou translucides ;
- possibilité de rendu en couleurs indexées (une couleur par sommet) ;
- affichage en utilisant les *vertex array* ou les VBO ;
- gestion d'un nombre quelconque de niveaux de détail ;
- possibilité de définir des groupes afin de les sélectionner ;
- possibilité de surcharger les propriétés de rendu.

Exemple des possibilités de rendu de GLC_lib

1.3.2. Structure hiérarchique d'une scène

Pour représenter la structure hiérarchique d'une scène, GLC_lib utilise « un graphe non orienté acyclique connexe » ou plus simplement un arbre.



Quantité : 1	Quantité : 1	Quantité : 4	Quantité : 2

La structure d'assemblage d'un quad (exemple fourni avec 3DXML Player)

Pour éviter de gaspiller de la mémoire, les géométries ainsi que les sous-ensembles qui apparaissent plusieurs fois sont instanciés. La définition d'un sous-ensemble ou d'une géométrie est appelée référence. La structure d'un assemblage est composée d'occurrences, d'instances et de références. Suivant cette architecture, le quad est composé des éléments suivants :

- quatre références de géométries : châssis, volant, roue et essieu ;
- deux références d'assemblages : assemblage du quad complet et assemblage essieu ;
- cinq instances de géométries : châssis.1, volant.1, roue.1, roue.2 et essieu.1 ;
- trois instances d'assemblage : quad, (essieu assemblé).1 et (essieu assemblé).2 ;
- huit occurrences de géométries ;
- trois occurrences d'assemblages.

1.3.3. Optimisations

Afin de charger et de visualiser des scènes complexes le plus rapidement possible, GLC_lib utilise des techniques d'optimisation connues et éprouvées.

Les optimisations utilisées sont axées sur l'affichage d'une scène composée d'un grand nombre de géométries et non sur l'affichage d'une seule géométrie.

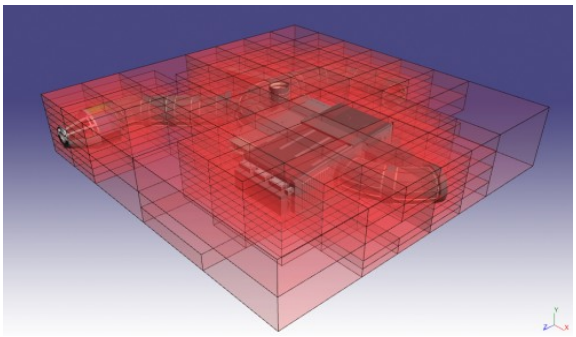
1.3.3.1. Chargement

Uniquement pour le format 3DXML de Dassault Systèmes, GLC_lib utilise une technique de cache qui consiste à créer un cache au format binaire des géométries à afficher. Ainsi, pour les chargements ultérieurs, le cache est utilisé et permet de diviser le temps de chargement d'un facteur compris entre cinq et dix.

1.3.3.2. Visualisation

Pour permettre l'affichage d'une scène complexe, le moteur de rendu de GLC_lib utilise les trois techniques suivantes :

- exclusion de pixels : élimination des objets dont la taille projetée est inférieure à un seuil exprimé en pixels ;
- niveaux de détail : remplacement par des objets ayant moins de triangles ;
- élimination des objet hors champ : élimination des objets situés en dehors du champ de vision de la caméra.



Partitionnement par Octree utilisé pour accélérer l'élimination des objets hors champ

1.4. Formats supportés

GLC_lib supporte les six formats suivants :

- Collada V1.4 (lecture seulement) ;
- 3DXML (XML) V3 and V4 (lecture et écriture en 3DXML V4) ;
- OBJ (lecture seulement) ;
- 3DS (lecture seulement) ;
- STL (ASCII et binaire) (lecture seulement) ;
- OFF et COFF (lecture seulement).

Le format de prédilection de GLC_lib est le 3DXML de Dassault Systèmes.

Ce format a été créé par Dassault Systèmes pour concurrencer les formats X3D et U3D.

Le 3DXML est décliné en trois variantes :

- exact : la géométrie est décrite en mode exact (B-Rep) et est encodée en binaire ;
- maillage : la géométrie est décrite en mode maillage et est encodée en ASCII ;
- maillage compressé : la géométrie est décrite en mode maillage et est encodée en binaire.

La variante encodée en ASCII a été rendue publique le 15 juin 2005. Cependant, il faut faire une demande auprès de Dassault Systèmes pour accéder à la spécification du 3DXML.

Le 3DXML est un conteneur compressé au format « ZIP » qui contient principalement un fichier XML décrivant la structure du produit et un fichier XML par pièce avec niveaux de détail.

Tous les logiciels de Dassault Systèmes permettent de lire ou d'exporter en 3DXML. Pour étendre l'utilisation de 3DXML, Dassault fournit gratuitement les logiciels 3DVIA et 3DXML Player ainsi qu'un site de partage de modèles 3D au format 3DXML et COLLADA.

2. Compilation et installation de GLC_lib

2.1. Présentation et prérequis

GLC_lib utilise l'API OpenGL et est très fortement couplée au framework Qt4 de Nokia. Pour compiler GLC_lib, il faut donc disposer d'une machine supportant l'OpenGL sur laquelle est installé Qt4. Comme GLC_lib n'a pas d'autre dépendance, ce sont les seuls prérequis.

À moins que vous souhaitiez modifier le code de GLC_lib, je recommande de ne pas utiliser d'EDI pour sa compilation et son installation. Pour la compilation d'une bibliothèque, il est plus complexe de paramétrer un EDI que de saisir trois lignes de commandes dans un interpréteur de commande.

Avant la compilation, il faut s'assurer que Qt4 est correctement installé en compilant un des exemples de Qt4 utilisant OpenGL (« Hello GL » me paraît être un bon candidat).

Ensuite, il faut télécharger et extraire les sources de GLC_lib : Lien 77.

Ne pas extraire le contenu de l'archive dans un dossier dont le chemin comporte des espaces ou des caractères spéciaux, beaucoup de compilateurs ne le supportent pas.

2.2. Les plateformes compatibles

Théoriquement, GLC_lib peut être compilée sur toutes les plateformes supportées par Qt4 prenant en charge OpenGL.

Voici la liste des plateformes testées :

- Windows Vista et Windows 7 en 32 bits avec MinGW32 ;
- Windows 7 en 64 bits avec MSVC 2008 ;
- Mac OS X 10.5, 10.6 ;
- Linux Ubuntu et Fedora en 32 et 64 bits.

Pour les plateformes Win32, il existe une version déjà compilée sur la page de téléchargement de GLC_lib : Lien 77.

Sur les plateformes Linux, il existe des paquets pour les architectures 32 bits et 64 bits sur le site : pkgs.org (Lien 78)

2.3. Compilation et installation sous Windows

Si vous avez installé le SDK de Qt : lancez l'interpréteur de commandes de Windows via le raccourci du SDK de Qt. Si vous utilisez MSVC++, lancez l'interpréteur de commande de Windows via le raccourci de MSVC++.

L'exemple est donné pour la version 2.1.0 de GLC_lib.

Allez dans le répertoire contenant les sources de GLC_lib :

```
C:\Qt\2010.05\qt> cd \
C:\>cd compilation\GLC_lib_src_2.1.0\glc_lib
```

Tapez la ligne de commande suivante pour générer le « *makefile* » :

```
C:\compilation\GLC_lib_src_2.1.0\glc_lib> qmake
```

2.3.1. MinGW

Tapez la ligne de commande suivante pour compiler GLC_lib :

```
C:\compilation\GLC_lib_src_2.1.0\glc_lib>
mingw32-make
```

Tapez la ligne de commande suivante pour installer GLC_lib :

```
C:\compilation\GLC_lib_src_2.1.0\glc_lib>
mingw32-make install
```

2.3.2. Visual C++

Tapez la ligne de commande suivante pour compiler GLC_lib :

```
C:\compilation\GLC_lib_src_2.1.0\glc_lib> nmake
```

Tapez la ligne de commande suivante pour installer GLC_lib :

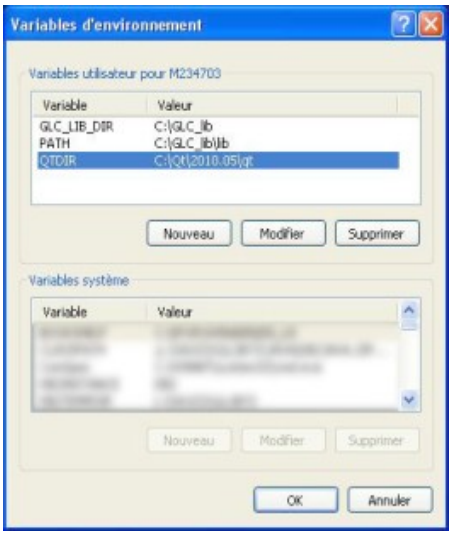
```
C:\compilation\GLC_lib_src_2.1.0\glc_lib> nmake
install
```

2.3.3. Fin d'installation

Par défaut, GLC_lib est installée dans le dossier C:\GLC_lib qui contient :

- un dossier « lib » pour la DLL et le fichier d'importation des symboles (.lib) ;
- un dossier « include » pour les en-têtes.

Pour compléter l'installation, il reste à ajouter le dossier C:\GLC_lib\lib dans le PATH et à définir la variable « GLC_LIB_DIR » avec la valeur : C:\GLC_lib.



Vous pouvez maintenant utiliser la bibliothèque dans vos programmes.

2.4. Compilation et installation sous Unix (Linux, BSD, Mac OS X, etc.)

Pour pouvoir utiliser les outils de développement sous Linux, vous devez installer les packages nécessaires pour le développement d'applications Qt4.

Après l'installation de ces packages, lancez une fenêtre terminal et allez dans le répertoire contenant les sources de GLC_lib :

```
cd /Users/laumaya/GLC_lib_src_2.1.0/glc_lib
```

Sous les autres systèmes que Mac OS X : tapez la ligne de commande suivante pour générer le « *makefile* » :

```
qmake
```

Sous Mac OS X, tapez la ligne de commande suivante pour générer le « *makefile*» :

```
qmake -spec macx-g++
```

Tapez la ligne de commande suivante pour compiler GLC_lib :

```
make
```

Tapez la ligne de commande suivante pour installer GLC_lib et tapez votre mot de passe

```
sudo make install
```

La bibliothèque est installée dans /usr/local/lib

Les fichiers d'en-têtes sont installés dans /usr/local/include/GLC_lib

3. Documentation de GLC_lib

Je suis conscient que le gros point faible de GLC_lib est sa documentation peu fournie.

C'est d'ailleurs la raison qui m'a poussé à écrire cet article.

Cependant, il existe plusieurs sources de documentation de GLC_lib qui sont principalement en anglais.

Lien 79

3.1. Aide sur le site

Lien 80

Sur le site Web de GLC_lib, vous pouvez trouver :

- une aide concise sur la compilation et l'installation de GLC_lib ;
- la documentation de référence ;
- des exemples d'utilisation dans l'ordre croissant de leur complexité.

3.2. Documentation de référence

Lien 81

Cette documentation a été générée par Doxygen.

3.3. Forum

Si vous avez besoin de plus d'aide sur GLC_lib, vous pouvez utiliser les forums dédiés à GLC_lib.

Version en français : Lien 82

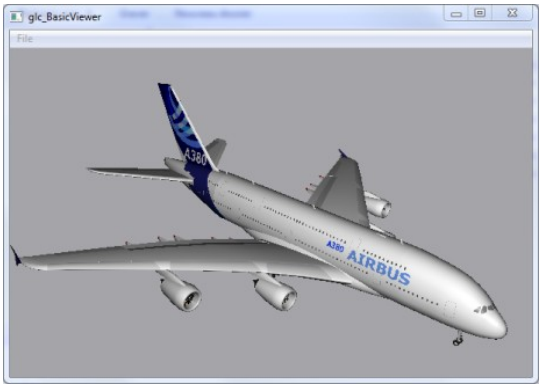
Version en anglais : Lien 83

4. Exemple d'utilisation de GLC_lib

Pour cet exemple, nous allons créer un logiciel permettant de visualiser tous les formats 3D supportés par GLC_lib.

Code source de l'exemple : Lien 84

Logiciel de visualisation GLC_BasicViewer (Mac)



Logiciel de visualisation GLC_BasicViewer (Windows)

4.1. Création du projet (le fichier .pro)

Code 1 : Lignes à ajouter dans le fichier projet d'une application utilisant GLC_lib

```
win32 {
    LIBS += -L"$$(GLC_LIB_DIR)/lib" -lGLC_lib2
    INCLUDEPATH += "$$(GLC_LIB_DIR)/include"
}

unix {
     LIBS += -lGLC_lib
     INCLUDEPATH += "/usr/include/GLC_lib"
}
```

Pour utiliser GLC_lib, il faut lier le projet à la bibliothèque.

Sous Windows :

```
LIBS += -L"$$(GLC_LIB_DIR)/lib" -lGLC_lib2
```

GLC_LIB_DIR est la variable d'environnement définissant

le chemin d'installation de GLC_lib.

Sous Unix (Linux et Mac) :

```
LIBS += -lGLC_lib
```

Comme GLC_lib est installée par défaut dans /usr/local/lib, il n'est pas nécessaire de spécifier son emplacement.

Ensuite, il faut définir l'emplacement des en-têtes de GLC_lib :

Sous Windows :

```
INCLUDEPATH += "$$(GLC_LIB_DIR)/include"
```

Sous Unix (Linux et Mac) :

```
INCLUDEPATH += "/usr/local/include/GLC_lib"
```

Code 2 : le fichier projet de l'exemple

```
TEMPLATE = app
QT += opengl

win32 {
    LIBS += -L"$$(GLC_LIB_DIR)/lib" -lGLC_lib2
    INCLUDEPATH += "$$(GLC_LIB_DIR)/include"
}

unix {
     LIBS += -lGLC_lib
     INCLUDEPATH += "/usr/include/GLC_lib"
}

# Input
HEADERS += glwidget.h mainwindow.h
SOURCES += glwidget.cpp main.cpp mainwindow.cpp
```

4.2. Description de l'application

Pour cet exemple, nous allons créer une application composée d'une fenêtre principale héritant de QMainWindow ayant pour Widget central une classe héritant de QGLWidget.

4.2.1. Code source de la classe MainWindow

La classe MainWindow contient uniquement deux méthodes :

- son constructeur ;
- un SLOT déclenché lors de l'ouverture d'un fichier.

4.2.1.1. Le constructeur de la classe MainWindow

Code 3 : MainWindow::MainWindow()

```
MainWindow::MainWindow()
: QMainWindow()
, m_GLWidget(this) // La vue Opengl (QGLWidget)
, m_pOpenFileAction(new QAction(tr("Open"),
this))
, m_CurrentPath(QDir::homePath())
{
        setCentralWidget(&m_GLWidget);
        // Création du menu
        QMenu* pMenu= new QMenu(tr("File"),
this);
        pMenu->addAction(m_pOpenFileAction);
        this->menuBar()->addMenu(pMenu);

        // Connexion de l'action d'ouverture de
fichier au SLOT void open()
        connect(m_pOpenFileAction,
SIGNAL(triggered()), this, SLOT(open()));
}
```

Pour le constructeur de cette classe, il n'y a pas de code relatif à GLC_lib.

4.2.1.2. La méthode d'ouverture de fichier

Code 4 : void MainWindow::open()

```
void MainWindow::open()
{
  // Liste des formats de fichiers ?? filtrer
  QStringList filters;
  ....

  const QString message(tr("Select File(s) to
Open and Add in album"));
  QString fileName =
     QFileDialog::getOpenFileName(this, message,
m_CurrentPath, filters.join("\n"));
  if (!fileName.isEmpty())
  {
    // Affichage du sablier
    QApplication::setOverrideCursor(QCursor(Qt::W
aitCursor));

    // Modification du répertoire courant
    m_CurrentPath=
QFileInfo(fileName).absolutePath();

    QFile file(fileName);

    // Construction de la scène 3D à partir du
fichier
    GLC_World world= GLC_Factory::instance()->
createWorldFromFile(file);

    if (!world.isEmpty())
    {
      m_GLWidget.setWorld(world);
    }

    // Restauration du curseur
    QApplication::restoreOverrideCursor();
  }
}
```

Nous avons deux actions importantes dans ce code

- Lecture du fichier et création de la scène 3D :

```
// Construction de la scène 3D à partir du
fichier
GLC_World world= GLC_Factory::instance()->
createWorldFromFile(file);
```

On accède à l'unique instance de la classe GLC_Factory par sa méthode statique instance(), ensuite la « Factory » s'occupe de construire la classe de lecture appropriée pour le fichier passé en paramètre. Le type de fichier est déterminé par son extension. Si tout se passe bien, la scène 3D correspondant au fichier est renvoyée.

Pour simplifier cet exemple, les exceptions susceptibles d'être levées lors de la lecture du fichier ne sont pas interceptées.

- Passage de la scène à la vue :

```
m_GLWidget.setWorld(world);
```

La classe GLC_World partage ses données de façon implicite. Sa copie est donc très rapide et la destruction de sa dernière instance libérera la mémoire allouée comme pour un pointeur intelligent.

4.2.2. Code source de la classe GLWidget

Cette classe est le coeur de notre application.

4.2.2.1. Le constructeur de la classe GLWidget

Code 5 : GLWidget::GLWidget(QWidget *p_parent)

```
GLWidget::GLWidget(QWidget *p_parent)
: QGLWidget(p_parent)
, m_Light()                        // Lumière
(Omnidirectionnelle) : GLC_Light
, m_World()                        // La scène 3D :
GLC_World
, m_GlView(this)                   // La vue OpenGL
GLC_Viewport
, m_MoverController()   // Le contrôleur des
interactions lié au point de vue
GLC_MoverController
{
  // Définition de la position de la lumière
  m_Light.setPosition(1.0, 1.0, 1.0);

  // Création du contrôleur des interactions par
défaut
  QColor repColor;
  repColor.setRgbF(1.0, 0.11372, 0.11372, 1.0);
// Couleur des éléments
  m_MoverController= GLC_Factory::instance()->
createDefaultMoverController(repColor,
&m_GlView);
}
```

Le constructeur se charge de créer et d'initialiser quatre objets de GLC_lib. Parmi ces objets, le plus intéressant est celui de la classe GLC_MoverController. Cet objet permet de gérer les interactions utilisateur via des évènements de la souris. La classe GLC_Factory construit un contrôleur de navigation par défaut.

4.2.2.2. L'initialisation OpenGL

Code 6 : void GLWidget::initializeGL()

```
void GLWidget::initializeGL()
{
        m_GlView.initGl();
        m_GlView.setBackgroundColor(Qt::gray);

        GLC_State::setVboUsage(false);
}
```

Lors de l'initialisation OpenGL, la première chose à faire est d'appeler la méthode : GLC_Viewport::initGl(). Cette méthode effectue les actions suivantes :

- initialisation des états d'OpenGL requis par GLC_lib ;
- chargement des extensions OpenGL ;
- initialisation des états de GLC_lib.

Ensuite, on définit la couleur de l'arrière-plan et on désactive l'extension OpenGL « Vertex Buffer Object » qui peut poser des problèmes sur certains types de matériel.

4.2.2.3. Le redimensionnement de la fenêtre OpenGL

Code 7 : void GLWidget::resizeGL(int width, int height)

```
void GLWidget::resizeGL(int width, int height)
{
        m_GlView.setWinGLSize(width, height);
}
```

La méthode GLC_Viewport::setWinGLSize(int, int) permet de mettre à jour les états OpenGL liés à la taille de la fenêtre OpenGL.

4.2.2.4. Assignation de la scène 3D à afficher

Code 8 : void GLWidget::setWorld(const GLC_World& world)

```
void GLWidget::setWorld(const GLC_World& world)
{
  m_World= world;
  if (!m_World.boundingBox().isEmpty())
  {
    m_GlView.reframe(m_World.boundingBox()); //
Recadrage de la vue sur la scène
    updateGL();         // Mise à jour de la vue
  }
}
```

Après l'assignation de la scène 3D à afficher, on recadre la vue sur celle-ci, puis on rafraîchit la vue.

4.2.2.5. Dessin de la vue OpenGL

Code 9 : void GLWidget::paintGL()

```
void GLWidget::paintGL()
{
  // Effacement de la vue
  glClear(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);

  // Chargement de ma matrice identité
  glLoadIdentity();

  // Calcul de la profondeur de champ de la scéne
  m_GlView.setDistMinAndMax(m_World.boundingBox()
);

  // Eclairage de la scène
  m_Light.enable();
  m_Light.glExecute();

  // Modification de la matrice de visulisation
en fonction
  // de la position de la caméra
  m_GlView.glExecuteCam();

  // Rendu de la scène
  m_World.render(0, glc::ShadingFlag);
  m_World.render(0, glc::TransparentRenderFlag);

  // Rendu du manipulateur de navigation courant
  m_MoverController.drawActiveMoverRep();
}
```

On remarquera que la scène est rendue en deux passes. La

première pour les faces opaques et la deuxième pour les faces translucides.

4.2.2.6. Réaction à l'appui sur un bouton de la souris

Code 10 : void GLWidget::mousePressEvent (QMouseEvent *e)

```
void GLWidget::mousePressEvent(QMouseEvent *e)
{
  if (m_MoverController.hasActiveMover()) return;
  switch (e->button())
  {
  case (Qt::RightButton):
    m_MoverController.setActiveMover(GLC_MoverCon
troller::TrackBall, e);
    updateGL();
    break;
  case (Qt::LeftButton):
    m_MoverController.setActiveMover(GLC_MoverCon
troller::Pan, e);
    updateGL();
    break;
  case (Qt::MidButton):
    m_MoverController.setActiveMover(GLC_MoverCon
troller::Zoom, e);
    updateGL();
    break;

  default:
    break;
  }
}
```

L'appui sur un des trois boutons de la souris permet de changer de mode de navigation courant :

- bouton droit : rotation ;
- bouton du milieu : déplacement panoramique ;
- bouton gauche : zoom .

4.2.2.7. Réaction au déplacement de la souris

Code 11 : void GLWidget::mouseMoveEvent (QMouseEvent * e)

```
void GLWidget::mouseMoveEvent(QMouseEvent * e)
{
  if (m_MoverController.hasActiveMover())
  {
    m_MoverController.move(e);
    updateGL();
  }
}
```

Grâce à la classe GLC_MoverController, la gestion des déplacements de la caméra se révèle très simple.

4.2.2.8. Réaction au relâchement d'un bouton de la souris

Code 12 : void GLWidget::mouseReleaseEvent (QMouseEvent*)

```
void GLWidget::mouseReleaseEvent(QMouseEvent*)
{
  if (m_MoverController.hasActiveMover())
  {
    m_MoverController.setNoMover();
    updateGL();
  }
}
```

Voici la dernière méthode de notre classe. Le relâchement d'un bouton de la souris enlève le mode de navigation courant.

5. Conclusion

J'espère que ce tutoriel vous aura donné l'envie d'utiliser GLC_lib.

Retrouvez l'article de Laurent Ribon en ligne : Lien 85

Gestion des risques en sécurité de l'information

Comment évaluer le risque pour les SI d'entreprise ? Quels risques doit-on accepter de prendre ? La gestion des risques en sécurité de l'information, recommandée par de nombreux référentiels comme la norme ISO 27001, est en train de devenir une obligation pour les responsables de la sécurité de l'information (RSSI), les directeurs des systèmes d'information (DSI) et bien d'autres acteurs de l'entreprise.
Après les démarches locales, comme Ebios et Mehari en France, la norme ISO 27005, première méthode de gestion des risques structurée et normalisée, est appelée à s'imposer à l'échelle planétaire. Facilement accessible, elle propose une approche continue (au quotidien, dans la durée), systématique, pragmatique et adaptée à la réalité complexe des entreprises actuelles. S'appuyant, tout comme la norme, sur des scénarios d'incidents réels, ce guide de mise en oeuvre ISO 27005 dévoile l'essence des années d'expérience et de savoir-faire de l'auteur et constituera une aide précieuse pour la certification ISO 27005 Risk Manager.

Critique du livre par Pierre Therrode

Évaluer les risques de sécurité dans une SI d'entreprise n'est pas une mince affaire lorsqu'il s'agit d'assurer les politiques de sécurité. Devons-nous fermer tous les accès qui relient une entreprise à l'Internet ou devons-nous accepter certains risques ?
Cet ouvrage est le premier livre en langue française à traiter la norme ISO 27005, écrit par Anne Lupfer avec une préface réalisée par Hervé Schauer, ce livre détaille chaque étape de la mise en place de la norme à travers bon nombre de schémas, d'exemples et de scénarii d'incidents. Il est vrai que certaines mises en application de la norme sont parfois délicates au premier abord, mais les exemples et les scénarii d'incidents sont là pour nous faire comprendre la mise en place de la norme ISO 27005.
Mais qu'est ce que la norme ISO 27005 ? En réalité, c'est une sorte de guide qui permet de définir les risques en sécurité de l'information. Cette norme a fait l'objet d'un standard international et apporte un renouveau dans la gestion des risques (anciennement ISO 27001), du fait qu'elle prenne en compte les risques liés au temps.
C'est donc au travers de différents chapitres tels que « l'identification des risques », « mesure de la portées des risques », « sélections des solutions » et de nombreuses études de cas que le lecteur en apprendra d'avantage sur le contenu de cette norme.
En conclusion, cet ouvrage s'adresse essentiellement aux responsables de la sécurité des systèmes d'information (RSSI), ainsi qu'au personnel impliqué dans la gestion et la mise en ouvre d'audits de sécurité.

Retrouvez cette critique de livre sur la page livres Sécurité : Lien 86

Les dernières news

Le premier virus sur PC a 25 ans

L'éditeur F-Secure retrace son histoire en vidéo

Le premier virus ayant infecté un PC a été découvert en 1986. Et, curieusement pour aujourd'hui, ce virus contenait les contacts de ses auteurs au Pakistan.

À l'occasion de cet anniversaire, Mikko Hyppönen, directeur du laboratoire de recherche de F-Secure, s'est donc rendu dans la ville de Lahore au Pakistan pour retrouver ces créateurs, deux frères, Amjad et Basit Farooq Alvi, qui sont aujourd'hui à la tête d'un FAI florissant (Brain Telecommunication Ltd.).

F-Secure propose un reportage vidéo de dix minutes sur ce voyage. Dans ce petit film nommé « Brain – à la recherche du premier virus sur PC », les deux frères donnent pour la première fois une interview sur leur virus qui s'est répandu à l'époque à travers le monde via des disquettes. Amjad et Basit y révèlent, par exemple, que Brain n'avait pas été conçu initialement pour être un virus de destruction, ce qui explique pourquoi ils avaient inclus leurs numéros de téléphone dans son code.

```
Welcome to the Dungeon
(c) 1986 Basit & Amjad (pvt) Ltd.
BRAIN COMPUTER SERVICES
730 NIZAB BLOCK ALLAMA IQBAL TOWN
LAHORE-PAKISTAN
PHONE :430791,443248,280530.
Beware of this VIRUS....
Contact us for vaccination..... $#@%$ @
```

Leur but était en fait de vérifier la fonctionnalité multitâche du nouveau système d'exploitation DOS et de voir si celui-ci contenait des failles de sécurité plus ou moins importantes que les autres systèmes d'exploitation comme Unix. Un PoC détourné de son objectif initial, donc.

Mikko Hypponen cache mal son excitation d'enfant lors de cette enquête : *« c'était incroyable. Je suis allé au Pakistan, à l'adresse notée dans le code du virus. J'ai frappé à la porte et les deux frères qui ont crée Brain - il y a 25 ans - m'ont ouvert »*, raconte-t-il. *« Lorsqu'Amjad et Basit ont écrit ce virus en 1986, le monde était très différent. Il n'y avait aucune intention malveillante lors de la création de ce qui fut le premier virus sur PC. Je pense que nous avons enregistré une part importante de l'histoire de l'informatique dans cette vidéo. »* : Lien 87.

Brain partait donc d'une bonne intention. Mais le chemin des enfers est pavé de bonnes intentions, dit le dicton.

Commentez la news d'Hinault Romaric en ligne : Lien 88

Les derniers tutoriels et articles

Les Malwares pour les nuls

Ce petit dossier est à destination des débutants souhaitant acquérir le minimum de connaissances au sujet des malwares pour éviter de se retrouver avec un PC infecté, et de passer par la case désinfection ou réparateur.
Je vous conseille de le lire, il n'est pas compliqué du tout, vraiment court, et vous permettra d'éviter les infections si vous appliquez les conseils donnés.
PS : Le titre est un clin d'œil à la collection Pour les nuls, ça n'a absolument rien de péjoratif.

1. Présentation des malwares

1.1. Définition

Un logiciel malveillant *(en anglais,* ***malware****) est un logiciel développé dans le but de nuire à un système informatique* (Wikipédia)

Grosso modo, le mot **Malware** est le mot qui désigne à lui seul tous les types de logiciels malveillants. Il permet donc de désigner des logiciels tels que des *Virus*, des *Vers*, des *Rogues*, des *Troyens* des *Rootkits*...
Tous les mots cités ci-dessus désignent une famille de malwares, un certain type de malwares des malwares ayant des caractéristiques communes.

1.2. Cœur du sujet

Au début de Internet, les malwares n'existaient pas au sens propre du terme. En effet seuls les virus et les vers informatiques existaient déjà.

Les rogues, rootkits, troyens, et toute la ribambelle d'autres familles de malwares n'existaient pas encore.

C'est pour cela, que **le mot malware** n'est que peu connu du fait qu'au début de l'ère Internet les virus étant les menaces les plus présentes, le mot *virus* était utilisé en permanence par les médias. Ce mot est donc resté pour désigner les menaces du net. Mais entre temps, les menaces ont évolué, elles se sont multipliées et diversifiées.

Le mot virus, est la majorité du temps utilisé pour désigner l'ensemble des menaces mais un virus à proprement dit est destructeur et infecte de nombreux fichiers sur le PC, ce sont des fichiers hôtes. Ce mot est un abus de langage donc désormais, quand vous parlerez des menaces informatiques, vous utiliserez l'expression *Logiciel Malicieux ou Malware*.

Mais de nos jours, bien qu'existent les mots "*troyens, virus, vers, rootkits...*" la majorité des infections qui

polluent nos PC sont en fait apparentées à plusieurs familles de malwares. Prenons pour exemple, l'infection nommée "*Navipromo*", cette infection est un adware c'est-à-dire qu'elle affiche des publicités intempestives et pourtant Navipromo utilise un rootkit pour dissimuler l'infection et permettre toutes sortes d'activités à l'insu de l'utilisateur et aux solutions de sécurité.

C'est pour cela qu'il faut faire attention aux **termes employés** et utiliser (s'il existe) le nom de l'infection elle-même cela évite de se mélanger les pinceaux et de dire ou de faire n'importe quoi.
Comme nous l'avons rapidement vu, le début de l'ère des malwares, s'annonce par l'apparition de virus et de vers au début de la création de l'informatique, et lors de l'apparition d'Internet et des premiers échanges par mail. Donc au début, les seules menaces se résumaient aux vers et aux virus, qui permettaient à leurs créateurs de démontrer leurs prouesses techniques et de prouver au "commun des mortels", qu'ils pouvaient détruire un PC, faire tomber des serveurs...

Les malwares ont évolué ils ne servent plus à détruire des systèmes ou à tenter d'infecter le plus de gens possible dans le seul but que l'on parle d'eux. Non, aujourd'hui les malwares deviennent de plus en plus discrets, et leurs créateurs n'ont qu'une peur, que l'on parle d'eux, et que l'on s'intéresse à eux. Cela est en totale opposition avec les anciens pirates informatiques. Les pirates contemporains désirent rester dans l'ombre. Mais pourquoi tiennent-ils à rester cachés ?
Ils veulent rester cachés, car leurs motivations et leurs buts ont changé. Le seul but des pirates aujourd'hui et de générer du profit, de leur rapporter de l'argent en infectant le plus grand nombre de machines en un minimum de temps. Voilà quel est le but des pirates du XXIe siècle. Et c'est pour cela que des technologies comme les systèmes de **rootkits**, des malwares comme les **adwares** ou les **rogues** ont vu le jour, car certains permettent de générer de l'argent et les autres de rester discrets. Nous verrons dans une autre page comment les pirates gagnent de l'argent avec leurs créations.

Je voudrais aussi vous parler de l'apparition de ce que l'on nomme des **Botnets** (roBOT NETwork) ou en français des **PC zombies**.
Définition :
En Sécurité informatique, une machine zombie est un ordinateur contrôlé à l'insu de son utilisateur par un pirate informatique, ce dernier l'utilise alors le plus souvent à des fins malveillantes.

Un botnet est donc en **ensemble de PC reliés entre eux** par des chevaux de Troie, et tous ces ordinateurs sont contrôlés par un pirate informatique qui peut leur faire faire ce que bon lui semble. Comme **attaquer** les serveurs d'une entreprise et leur demander une somme d'argent en échange de l'arrêt de l'attaque ou encore envoyer 10 000 000 de spams à différentes adresses mail en échange d'une somme d'argent donnée par un autre pirate, qui paye pour utiliser le botnet. Ou encore infecter 5000 ordinateurs d'adware pour afficher des pubs d'un site de casino en échange d'une somme d'argent donnée par ce dernier...

Ce qu'il faut retenir de ceci, c'est que **les malwares évoluent, se diversifient**, et cette évolution n'est pas terminée et est de plus en plus inquiétante.

2. Les différents malwares

Comme vous le voyez il existe de très nombreux malwares. Comme je l'explique en dessous des définitions, de nombreuses infections rentrent dans plusieurs catégories.
Voici les **différents types de malwares existants** :

Adware/Publiciel	Logiciel affichant des publicités
Backdoor/Porte dérobée	Logiciel permettant l'accès à distance d'un ordinateur de façon cachée.
Bot	Logiciel automatique qui interagit avec des serveurs.
Exploit	Logiciel permettant d'exploiter une faille de sécurité.
Keyloger/Enregistreur de frappe	Logiciel permettant d'enregistrer les touches frappées sur le clavier.
Ransomware/Rançongiciel	Logiciel qui crypte certaines données du PC, et demande une rançon pour permettre le décryptage.
Rogue	Logiciel se faisant passer pour un antivirus, et indiquant que le PC est gravement infecté. Il se propose de le désinfecter en échange de l'achat d'une licence.
Rootkit	Logiciel permettant de cacher (et de se cacher lui-même) une infection sur un PC.
Spammeur	Logiciel envoyant du spam/pourriel.
Spyware/espiogiciel	Logiciel collectant des informations sur l'utilisateur.
Trojan horse /Cheval de Troie	Logiciel permettant la prise de contrôle à distance d'un PC, il permet souvent l'installation d'une porte dérobée.
Ver/Virus réseau	Logiciel se propageant via un réseau informatique.
Virus	Logiciel conçu pour se propager de PC en PC et s'insérant dans des programmes hôtes.

Il existe donc de **très nombreuses menaces**. Aujourd'hui, les limites entre les différentes sortes de malwares, programmes malicieux, et autres menaces sont de plus en plus floues et difficiles à définir. En effet, il fut un temps, où nous pouvions facilement désigner une menace avec

telle ou telle dénomination. Aujourd'hui une seule menace peut obtenir plusieurs dénominations, tout simplement parce qu'elle rentre dans les critères de plusieurs familles et constitue en fait **une association de malwares**.

Certains d'entre eux, sont des Virus, mais remplissent aussi la fonction de keylogger. On peut donc penser que tous ces noms sont faussés, et qu'ils ne veulent plus dire grand-chose. Faux : ces mots permettent tout de même de cerner le type de stratégie et les capacités de contamination du malware auquel on a affaire, et l'on peut donc, via sa dénomination, déjà deviner ses aptitudes.
Bref, à quoi il sert, ce qu'il fait.

3. Les autres nuisances informatiques

En dehors des malwares, il existe d'autre sortes de nuisances informatique.

L'ingénierie sociale, technique de plus en plus utilisée par les pirates pour pousser les internautes à s'infecter, permet aussi de réaliser ce que l'on appelle le phishing ou hameçonnage : Lien 89.
Cette pratique, consiste à se faire passer pour X auprès de Y et de profiter de ce rôle pour soutirer des informations personnelles à Y.
Autrement dit, un pirate se fait passer pour une banque et demande par mail, via un faux mail bancaire des informations confidentielles à l'internaute, comme ses numéros de comptes, le mot de passe de son adresse e-mail ...

Sachez, que votre banque/FAI... ne vous demandera **jamais** ce genre d'informations par Internet, donc si vous recevez ce genre de mails effacez les tout de suite de votre courrier.

Toutes les autres attaques informatiques comme les attaques DdoS (Lien 90) sont des nuisances informatiques, mais ne concernent que très peu les utilisateurs lambda, donc nous ne nous attarderons pas dessus.

4. A quoi servent les malwares ?

C'est vrai ça, à quoi servent-ils ? A qui tout cela profite-t-il ?

Les malwares depuis quelques années servent à générer de l'argent, et donc à enrichir ceux qui les créent.

Je vais prendre des exemples pour montrer comment gagner de l'argent grâce aux malwares pour ceux qui auraient du mal à voir comment cela fonctionne.
Les chiffres exposés ici sont pris totalement **au hasard**, ils sont **fictifs**.

4.1. Premier exemple

Prenons l'exemple d'un Adware qui affiche des publicités sauvages pour un site de paris en ligne, de commerce en ligne ou de jeux (poker).
Admettons que cet adware infecte 5000 PC, et qu'il affiche ses pop-ups une fois par jour. Environ 5000 personnes verront donc cette pub pour ce site (sûrement plus car il y a bien souvent plusieurs utilisateurs par PC), combien cliqueront dessus ?

Disons que 3000 cliquent sur la pub et donc se rendent sur le site. Il y a donc, grâce à l'adware, 3000 personnes qui se sont rendues sur le site, ce qui permet au webmaster de ce dernier d'augmenter le nombre de personnes qui visitent son site. De ce fait il va gagner encore plus d'argent grâce aux pubs présentes sur son site, car elles seront vues par encore plus de gens.

Estimons maintenant que 1500 personnes cliquent sur les pubs présentes sur le site de paris, cela augmente encore l'argent gagné par le webmaster, car 1500 personnes en plus auront cliqué grâce à l'adware sur les pubs du site.
Le créateur de l'adware a donc permis au webmaster du site de paris non seulement d'augmenter le nombre de personnes qui visitent le site, mais en plus d'augmenter le nombre de personnes qui cliquent sur les pubs présentes dessus.
Ceci a donc permis au webmaster de faire du profit. Il va donc à son tour donner de l'argent au créateur de l'adware pour le remercier de son service.

Voilà un cas de figure de la création d'adwares. Cela marche de façon similaire pour les autres types d'infections et pirates qui créent des rogues, des trojans, des keyloggeurs ou des rootkits...

Comme vous le voyez, il est très simple pour les pirates de gagner de belles sommes d'argent en profitant de la naïveté des internautes. Et c'est pour cette simple raison que les malwares existent, et existeront sûrement toujours, puisqu'ils génèrent de l'argent, beaucoup d'argent, et dans le monde actuel, ou du moins dans le système où le monde se trouve, l'argent contrôle tout, il est donc normal que le nombre de pirates augmente.

4.2. Deuxième exemple

Cette fois prenons le cas d'un pirate qui vient de réaliser un keyloggeur qui permet de récupérer les mots de passe personnels et les numéros de carte bancaire, et qui l'a associé à un rootkit pour détourner l'attention des antivirus.

Ce pirate infecte via son malware 10 000 PC (plus que l'adware car il est plus discret, donc plus difficile à repérer, et logiquement plus long à supprimer). Sur ces 10 000 PC il récupère 9000 numéros de carte bancaire, et 15 000 mots de passe de boite de messagerie distante en tout genre (Hotmail, Yahoo, SFR, Orange...).
Le pirate récupère tout ça, et se rend sur un forum underground où il annonce à tous les autres utilisateurs du forum qu'il détient de nombreux numéros de carte bleue et mots de passe.

De nombreuses personnes le contactent pour acheter des numéros de carte bleue et des mots de passe d'adresse mail qu'il vend disons 5 euros l'un. Des pirates spécialisés dans le spam le contactent pour lui demander des mots de passe d'adresses mail.

La question est : **pourquoi les gens achètent-ils des numéros de carte bleue et des mots de passe de boîtes mails ?**
Pour les cartes bleues, facile, cela leur permet de faire des achats en ligne.

Et pour les mots de passe, cela leur permet de spammer les contacts de l'adresse mail dont le mot de passe a été volé. Spams qui renverront le plus souvent vers des sites qui augmenteront ainsi leur nombre de visites, et donc leurs gains.

C'est un cercle vicieux, chacun des acteurs a besoin de l'autre, et ce système repose sur l'argent.
Je souligne aussi que de nombreux pirates n'ont pas créé les malwares qu'ils utilisent, mais les ont achetés à d'autres pirates qui les développent puis les vendent.

Donc *l'art* de l'infection peut être réalisé par des néophytes en la matière ce qui ouvre encore plus grand le monde du piratage, et augmente encore plus cette menace croissante.
Le but des pirates et donc de gagner de l'argent, c'est l'évidence même. Autrement dit, lorsque vous ne protégez pas votre PC vous faites gagner de l'argent aux pirates et dévoilez votre vie privée.

4.3. L'organisation des pirates

Certains pirates s'organisent parfois pour par exemple monter un botnet ou réaliser un ver comme *Storm* qui a permis d'infecter des milliers et des milliers de PC.
Ces réseaux de pirates s'organisent comme des TPE, chacun des pirates a un rôle bien défini dans l'organisation, il y a une professionnalisation du cybercrime.

Dans certains pays, les informaticiens n'ont pas de travail et sont forcés de se tourner vers le milieu underground pour survivre, ce qui peut expliquer l'augmentation du nombre de malwares. Ce sont ces informaticiens qui deviennent des pirates en travaillant avec des mafias organisées, qui ont vu dans le cybercrime un nouveau moyen de gagner de l'argent.

L'évolution des malwares, comme l'apparition et la banalisation de botnets, ou de rootkits, montrent bien que les pirates se tiennent à la page, innovent et cherchent des techniques pour permettre de gagner toujours plus d'argent.

N'est-il pas plus rentable d'installer un rootkit sur un PC, et qu'il y reste caché pendant plusieurs mois, que d'installer un adware qui sera supprimé en quelques semaines ?
Si, ça l'est, et c'est pour cela que les malwares vont devenir **de plus en plus furtifs et difficiles à déceler (infections du MBR...)**.

En résumé, on peut simplement dire que les malwares seront présents dans encore très longtemps et ont malheureusement souvent une longueur d'avance, **donc restez prudent**.

5. Règles de base pour éviter les infections

5.1. Logiciels de sécurité à jour

Les antivirus utilisent plusieurs techniques pour détecter les malwares, notamment la *signature* des malwares.
Plus ils ont de signatures, plus ils ont de chances de détecter des malwares, et donc plus ils sont efficaces. Il faut donc tenir son antivirus bien à jour, dès que la mise à jour sort, on l'installe. De même pour les firewalls.

Si vous utilisez d'autres logiciels de sécurité, **appliquez la même politique**. C'est très important de tenir ses logiciels de sécurité à jour.
En effet, même à jour ils ne sont pas efficaces à 100 %, donc s'ils ne le sont pas, ils ne sont pas efficaces du tout.
Il est préférable de télécharger les logiciels de sécurités (et d'ailleur tous les logiciels) sur le site de l'éditeur ou sur des sites fiable. Et surtout ne téléchargez pas les logiciels proposés par des publicitées.

5.2. Tenir le système d'exploitation (OS) (Windows, Mac...) à jour

But : combler ce qu'on appelle des *failles de sécurité*. Ces failles sont, si l'on peut dire, des sortes de "trous" dans le système d'exploitation par lesquels les malwares peuvent pénétrer. Ce sont des vulnérabilités du système qui vous mettent en danger. Plus votre OS est à jour, moins il y a de "trous", donc moins il y a de risques potentiels de se faire infecter.
Faites vos **mises à jour** dès qu'elles vous sont proposées.

5.3. Tenir tous les logiciels du PC à jour

Le système d'exploitation et les logiciels de sécurité sont les plus importants à tenir à jour.
Mais il faut que **tous les logiciels de votre PC soient à jour**. Les logiciels qui doivent être impérativement à jour sont : votre **navigateur** (Internet Explorer, Firefox, Opera, Google Chrome)... **Java** (s'il est sur votre PC) **Adobe Flash et Adobe Reader** (s'ils sont sur votre PC)...

En effet, ces logiciels sont les plus visés par les infections, et tout comme le système d'exploitation, ces quatre logiciels peuvent contenir des vulnérabilités s'ils ne sont pas à jour.

Retenez bien : votre antivirus, votre firewall, votre système d'exploitation, votre navigateur, Java et Adobe **doivent impérativement être mis jour**.

5.4. Une bonne attitude sur le net

Par bonne attitude, je parle d'éviter les pièges les plus flagrants du net, comme : le **P2P**, **les cracks**, **les sites warez**, **les sites pornographiques**... En effet, ces quatre choses sont les principaux vecteurs d'infection de la toile. Évitez-les le plus possible, et au mieux **bannissez-les**, oubliez-les, n'en approchez plus, et votre PC aura **plus de chances de rester propre**.

Ce qui est très important aussi, c'est de comprendre que les **antivirus, les firewalls et autres logiciels de sécurité n'assurent pas à eux seuls la sécurité du PC, c'est à vous de faire attention à votre comportement**.
Contrairement à ce que pense la très grande majorité des gens, les logiciels de sécurité ne sont pas suffisamment puissants pour éviter toutes les infections et permettre au PC de rester sain, il y a de nouvelles infections tous les jours et tant qu'elles ne sont pas reconnues et incluses dans les bases virales, elles peuvent sévir sans barrières.

Vous non plus, vous ne pourrez être maitre à 100 % de la sécurité de votre PC, mais vous l'assurerez sûrement mieux que 10 anti-machins et 5 anti-trucs, car c'est vous qui dirigez le PC, vous qui décidez d'avoir une

conduite risquée comme d'utiliser Emule... vous qui décidez de télécharger des cracks...

Donc en conclusion, assurer la sécurité de votre PC, c'est avoir une **utilisation prudente et consciente de l'outil informatique dans son ensemble**.
Les solutions de sécurité vous aident dans ce travail, mais c'est vous qui dirigez le tout.

5.5. Restez vigilant et critique sur le net

En dehors d'exclure certaines choses, comme les cracks ou le P2P qui sont illégaux, et en plus de très gros vecteurs d'infection, certaines choses sont légales et divertissantes,et donc attirent du monde, donc les pirates...
Je parle des réseaux sociaux (*Facebook, Skype, Twitter*), des messageries instantanées (*MSN*), sans oublier les supports amovibles (*disque dur externe, clé USB, baladeur MP3...*).
En effet, ces trois choses sont banales, connues de presque tous et utilisées de façon très fréquente, mais **elles constituent aussi un risque d'infection**.

De plus en plus de pirates utilisent les réseaux comme **Facebook** pour propager leurs malwares, prenons l'exemple de **Koobface**. Pour lutter contre ces infections via réseaux sociaux, faites attention aux messages que vous recevez, et ne soyez pas trop naïfs.

Les **messageries instantanées** sont aussi utilisées pour infecter les internautes. Qui n'a jamais eu de message envoyé par un de ses contacts avec une phrase aguicheuse, et un lien, qui se révélera infectieux. Nous sommes nombreux à l'avoir eu, et vous êtes trop nombreux à avoir cliqué dessus. Il est préférable d'indiquer à l'expéditeur des messages dus aux infections qu'il est infecté.
De même que pour les réseaux sociaux, il faut ici aussi pour lutter contre ces infections être critique et non naïf.

Et les **clés USB**, comment pourraient-elles être dangereuses ?
Simple, vous les branchez à un PC infectieux, l'infection se copie/colle sur votre clé, et dès que vous la branchez sur un deuxième PC, le PC est à son tour infecté.
Pour éviter ces infections qui se propagent grâce à ce que l'on appelle l'Autorun, il suffit de désactiver ce dernier sur votre PC, et de vacciner vos disques amovibles avec des logiciels comme USBFix.
s Exemple de vers se répendant aussi par disque amovible : Conficker (Lien 91)

Les pirates utilisent de plus en plus ce que l'on appelle **l'ingénierie sociale**, c'est-à-dire la technique agissant sur le facteur humain, et plus précisément sur la crédulité des gens. Les mails piégés comme le phishing sont la face apparente de l'ingénierie sociale.
De plus en plus les pirates développent cette technique, car ils ont bien compris que **bien peu d'internautes sont informés et sensibilisés aux dangers du net**.
Soyez donc **très critiques face à ce que vous voyez sur le net**, il vaut mieux être **légèrement paranoïaque plutôt que d'être inconscient**.

6. Liens annexes

Actualité du moment :
Attaque informatique contre Bercy : Lien 92
Actualité sur la sécurité informatique : Lien 93

Livres/Magazines sur la sécurité informatique :
Livre sur la sécurité informatique : Lien 94
MISC-n°41 "LA CYBERCRIMINALITÉ ...ou quand le net se met au crime organisé" : Lien 95

Articles traitant du même sujet :
Les pièges de l'Internet : Lien 96

Organisation de lutte contre les menaces informatique en France :
OCLCTIC documentation : Lien 97

Autres :
Les Hoax : Lien 98

Retrouvez l'article de Simon Sayce en ligne : Lien 99

Concevoir facilement un plugin Nagios en Perl

Ce tutoriel a pour but de vous expliquer en quelques lignes comment concevoir un plugin Nagios respectant les normes Nagios avec Perl.

1. Introduction

Nagios est un logiciel open source de supervision. Il existe plusieurs logiciels se basant sur lui pour superviser un réseau informatique. Pour avoir plus d'informations sur son utilisation ou installation, vous pouvez vous référer à ce tutoriel - Installation et configuration de Nagios pour débutants : Lien 100. Nagios étant assez configurable, il donne la possibilité de créer ses propres programmes (**plugins**) afin de tester le bon fonctionnement d'un serveur, d'un matériel, d'un service. Ce tutoriel vous expliquera comment concevoir un tel programme en utilisant le langage Perl.

2. Plugins Nagios

2.1. Qu'est-ce que Nagios ?

Comme expliqué ci-dessus, Nagios est un outil de supervision. Mais si vous lisez cet article, nous supposons que vous connaissez déjà cet outil et de ce fait, nous ne rentrerons pas dans les détails de son mode de fonctionnement.

2.2. Qu'est-ce qu'un plugin Nagios ?

Les plugins Nagios sont des programmes compilés (écrits en C, C++, ...) ou des scripts (écrits en Perl, Shell, ...), ou tout autre langage de programmation pouvant s'exécuter en ligne de commande. Ils permettent de vérifier l'existence ou l'état d'un service, l'état d'un serveur... puis génèrent un résultat que Nagios exploite pour effectuer une action de notifications, pour exécuter des gestionnaires d'événements, etc. L'avantage est de superviser tout ce que l'on veut du moment que l'on peut écrire le plugin : un service HTTP, FTP, la taille d'un répertoire, l'existence d'un fichier sur un serveur, la température d'un onduleur...

Lorsque vous installez Nagios, il vous est également demandé d'installer des plugins, qui vous permettent d'effectuer certaines tâches. Sachez qu'il existe d'autres sites qui proposent des plugins Nagios gratuits.

2.3. Fonctionnement d'un plugin Nagios

Tout plugin Nagios doit retourner au moins une ligne de texte (En fait, les plugins doivent retourner du texte en sortie. A partir de la version 3 de Nagios, les plugins peuvent optionnellement retourner plusieurs lignes de texte.) sur la sortie standard STDOUT et renvoyer un code de retour compris entre 0 et 3.

Nagios ne s'intéresse pas au contenu d'un plugin, il ne se base que sur les informations (code de retour, texte) qu'il a reçues pour déduire le bon fonctionnement de l'hôte ou du service que vous surveillez. Voici comment Nagios interprète ce code de retour.

Code de retour du plugin	Etat du service	Etat de l'hôte
0	OK (tout va bien)	UP
1	WARNING (le seuil d'alerte est dépassé)	UP ou DOWN/UNREACHABLE (Si l'option *use_aggressive_host_checking* est activée, les codes de retour 1 donneront un état DOWN ou UNREACHABLE pour l'hôte. Sinon, les codes de retour 1 donneront un état UP pour l'hôte.)
2	CRITICAL (le service a un problème)	DOWN/UNREACHABLE
3	UNKNOWN (impossible de connaître l'état du service)	DOWN/UNREACHABLE

Explication des codes retour

Vous comprenez maintenant pourquoi il est simple de concevoir un plugin, les règles à respecter sont : simplicité, efficacité et légèreté. Le texte retourné doit être court (car il y a une restriction de longueur (Nagios ne lit que les premiers 4KB des données qui seront retournées, mais cette limite est modifiable si besoin en modifiant la définition **MAX_PLUGIN_OUTPUT_LENGTH** dans le fichier include/nagios.h.in du code source et il faudra recompiler Nagios. N'en abusez pas !!)) mais explicatif. Voici un exemple de lignes de sortie issu de la documentation officielle (Lien 101).

- DISK OK - free space: / 3326 MB (56%);
- DISK OK - free space: / 3326 MB (56%); | /=2643MB;5948;5958;0;5968

La sortie doit être de type **$SERVICEOUTPUT$**. Pour en savoir plus, consulter la documentation officielle.

2.4. Plugin Nagios en Perl

Entrons dans le vif du sujet !! Il est plus simple de comprendre en utilisant un exemple.
Nous avons à notre disposition deux serveurs :

- le premier nommé **"gendarme"** qui, comme vous l'aurez compris, a pour but de superviser le second serveur ;
- le deuxième nommé **"suspect"** qui sera notre serveur à surveiller ;

Ecrivons 3 plugins dont les rôles seront les suivants :

1. vérifier l'existence d'un fichier suspect ;
2. vérifier qu'un fichier ne dépasse pas une taille fixée ;
3. vérifier qu'un processus quelconque tourne bien sur le serveur **"suspect"**.

La surveillance pourrait se faire localement, le principe est le même. Parlons maintenant Perl. Un programme Perl minimal doit toujours contenir ces quelques lignes de code :

```
#!/usr/bin/perl
#=============================================
# Auteur : djibril
# Date   : 02/03/2011 22:42:13
# But    : Exemple de plugin Nagios
#=============================================
use strict;
use warnings;


__END__
```

Nous allons maintenant utiliser deux méthodes pour écrire nos plugins.

2.4.1. Méthode simple

Pour illustrer un exemple de méthode simple, créons un premier plugin avec pour nom **check_file_exist**. Son but est de vérifier qu'un fichier passé en argument est toujours présent sur le serveur distant **"suspect"**. Si oui, tout est OK, si non, Nagios doit générer une erreur critique et nous alerter. Ce plugin retournera (vous l'aurez compris), la valeur 0 ou 2. Voici notre programme :

```
#=============================================
# Auteur : djibril
# Date   : 04/03/2011 09:16:24
# But    : Plugin vérifiant l'existence d'un
fichier
#=============================================
use strict;
use warnings;
use Getopt::Long;
use File::Basename;

my ( $fichier_a_verifier ) = ();
GetOptions( 'file|f=s' => \$fichier_a_verifier );
if ( not defined $fichier_a_verifier ) {
  print "Il manque l'argument --file ou -f\n";
  exit 0;
}

my $NOM_PROCESS = 'CheckFileExiste';
my $nom_fichier = basename  $fichier_a_verifier;
# Vérification de l'existence du fichier
if ( -e $fichier_a_verifier ) {
  print "$NOM_PROCESS OK - $nom_fichier exists";
  exit 0;
}
else {
  print "$NOM_PROCESS CRITICAL - $nom_fichier not
exists";
  exit 2;
}

__END__
```

On remarque sa simplicité. Il vérifie l'existence d'un fichier et retourne une phrase ainsi qu'un code retour. Pour le tester nous allons nommer ce plugin **check_file_exist** (sans l'extension **.pl**, mais on peut la garder). Nous le mettons sur le serveur "**suspect**" dans le répertoire **/usr/local/nagios/libexec** (nous supposons que vous avez installé NRPE et les plugins Nagios sur le serveur), le rendons exécutable (**chmod +x /usr/local/nagios/libexec/check_file_exist**). Voici un test en ligne de commande.

```
suspect #
/usr/local/nagios/libexec/check_file_exist -f
/home/toto
CheckFileExiste CRITICAL - toto not exists
suspect # echo $?
2
```

Le programme fonctionne bien. Nous avons testé l'existence du fichier **/home/toto** et il n'a pas été trouvé. Un code retour de **2** a bien été retourné. Essayons de le tester depuis le serveur **"gendarme"** comme le ferait **Nagios** via **NRPE**.
Pour cela, sur le serveur **"suspect"**, modifions le fichier **/usr/local/nagios/nrpe.cfg** et rajoutons la ligne

nrpe.cfg
```
command[check_file_exist]=/usr/local/nagios/libex
ec/check_file_exist -f $ARG1$
```

Nagios appellera le programme avec l'argument **-f**. Depuis le serveur **"gendarme"**, nous pouvons lancer la ligne de commande suivante :

```
/usr/local/nagios/libexec/check_nrpe -n -H
suspect_ou_IP_suspect -c check_file_exist -a
/home/toto
CheckFileExiste CRITICAL - toto not exists

ou

CheckFileExiste OK - toto exists
```

Nous avons pu constater que nous arrivons à tester ce programme à distance comme l'aurait fait Nagios. Pour un fonctionnement depuis l'interface Nagios, il suffit de définir un nouveau service dans le fichier de configuration de Nagios permettant de surveiller l'hôte **"suspect"**. Par exemple :

suspect.cfg
```
define service{
    use                     generic-service
```

```
    host_name               suspect
    service_description     TEST
    check_command           check_nrpe!
check_file_exist!/home/djibril/.xtalkrc
}
```

Bon, ce programme Perl est vraiment très très simple, je dirais même trop simple. Nous allons voir maintenant comment ré-écrire ce même programme de façon plus propre car ce dernier présente quand même quelques inconvénients !!

2.4.2. Méthode propre

Nous avons vu dans l'exemple ci-dessus qu'il est assez facile d'écrire un petit programme fonctionnel, mais ce dernier contient quelques petits défauts :

- la gestion des arguments est faite mais peut être améliorée ;
- le code n'a pas de version ;
- il ne contient aucune documentation pour son usage ;
- un bon plugin doit être capable de fournir une aide via l'option **--help** ;
- la gestion du type de sortie de code retour est un peu brute et ne rend pas le programme lisible et maintenable facilement, même si ce dernier est assez simple ;
- de plus, si vous avez lu d'autres documentations de Perl et les plugins Nagios, il est souvent demandé d'exporter des variables (pour la gestion des codes retour), ce que nous n'avons pas fait ici.

Un module CPAN permet de générer facilement des plugins Nagios Perl propres : c'est le module Nagios::Plugin (Lien 102). Il respecte toutes les recommandations du Nagios Plugin guidelines (Lien 103). Vous devez l'installer (lisez ce tutoriel pour savoir comment installer un module Perl : Lien 104) sur le serveur qui contiendra le plugin.

2.4.2.1. Plugin 1 - check_file_exist

Reprenons notre code minimal Perl, chargeons le module et définissons une version pour notre plugin.

```
#!/usr/bin/perl
#===============================================
# Auteur : djibril
# Date   : 03/03/2011 20:49:03
# But    : Exemple de plugin Nagios
#===============================================
use strict;
use warnings;
use Nagios::Plugin; # Chargement du module

use vars qw/ $VERSION /; # Version du plugin
$VERSION = '1.0';

__END__
```

Créons notre constructeur Nagios::Plugin en lui spécifiant le **nom** de notre **process**, son **usage**, la **version** du programme et éventuellement une **licence**.

```
my $LICENCE
  = "Ce plugin Nagios est gratuit et libre de
droits, et vous pouvez l'utiliser à votre
convenance."
  . ' Il est livré avec ABSOLUMENT AUCUNE
GARANTIE.';

my $plugin_nagios = Nagios::Plugin->new(
  shortname => 'Check bank file',
  usage     => 'Usage : %s',
  version   => $VERSION,
  license   => $LICENCE,
);
```

Définissons les arguments que l'on souhaite lire. Dans notre cas, nous voulons récupérer le fichier à tester via l'option **-f** ou **--file**. Il faut utiliser la méthode **add_arg**.

```
# Définition de l'argument --file ou -f pour
récupérer le fichier
$plugin_nagios->add_arg(
  spec     => 'file|f=s',
  help     => 'file to check',     # Aide au sujet
de cette option
  required => 1,                   # Argument
obligatoire
);
```

Activons l'analyse des arguments du programme (--help, -V, -f, ...).

```
# Activer le parsing des options de ligne de
commande
$plugin_nagios->getopts;
```

Maintenant, nous pouvons effectuer notre test sur le fichier !

```
my $fichier_a_verifier = $plugin_nagios->opts-
>file;
my $nom_fichier        = basename
$fichier_a_verifier;

# Vérification de l'existence du fichier
if ( $plugin_nagios->opts->verbose ) { print
"Test du fichier $nom_fichier\n"; }
if ( -e $fichier_a_verifier ) {
  if ( $plugin_nagios->opts->verbose ) { print
"Fichier existant\n"; }
  $plugin_nagios->nagios_exit( OK, "$nom_fichier
exists" );
}
else {
  if ( $plugin_nagios->opts->verbose ) { print
"Fichier inexistant\n"; }
  $plugin_nagios->nagios_exit( CRITICAL,
"$nom_fichier not exists" );
}
```

La méthode **nagios_exit** permet de sortir du programme avec le code retour et notre message au format Nagios. Vous avez remarqué que nous avons rajouté des **print** en testant les méthodes **opts->verbose**. En fait, cette subtilité permet de faire un débogage si besoin. Si vous exécutez le programme avec l'option **-v**, vous aurez l'affichage des print. Voici notre programme final :

```
#!/usr/bin/perl
#===============================================
# Auteur : djibril
# Date   : 04/03/2011 09:16:24
# But    : Plugin vérifiant l'existence d'un
fichier
#===============================================
use strict;
use warnings;

# Chargement du module
use Nagios::Plugin;
use File::Basename;

use vars qw/ $VERSION /;

# Version du plugin
$VERSION = '1.0';

my $LICENCE
  = "Ce plugin Nagios est gratuit et libre de
droits, et vous pouvez l'utiliser à votre
convenance."
  . ' Il est livré avec ABSOLUMENT AUCUNE
GARANTIE.';

my $plugin_nagios = Nagios::Plugin->new(
  shortname => 'Verification fichier',
  usage     => 'Usage : %s [-f <file> ou --file
<file>]',
  version   => $VERSION,
  license   => $LICENCE,
);

# Définition de l'argument --file ou -f pour
récupérer le fichier
$plugin_nagios->add_arg(
  spec     => 'file|f=s',
  help     => 'file to check',    # Aide au sujet
de cette option
  required => 1,                  # Argument
obligatoire
);

# Activer le parsing des options de ligne de
commande
$plugin_nagios->getopts;

my $fichier_a_verifier = $plugin_nagios->opts-
>file;
my $nom_fichier        = basename
$fichier_a_verifier;

# Vérification de l'existence du fichier
if ( $plugin_nagios->opts->verbose ) { print
"Test du fichier $nom_fichier\n"; }
if ( -e $fichier_a_verifier ) {
  if ( $plugin_nagios->opts->verbose ) { print
"Fichier existant\n"; }
  $plugin_nagios->nagios_exit( OK, "$nom_fichier
exists" );
}
else {
  if ( $plugin_nagios->opts->verbose ) { print
"Fichier inexistant\n"; }
  $plugin_nagios->nagios_exit( CRITICAL,
"$nom_fichier not exists" );
}
__END__
```

Avec ces quelques lignes de code simples et claires, voici ce que nous obtenons en ligne de commande pour un test en local ou à distance :

```
suspect:~ #
/usr/local/nagios/libexec/check_file_exist
Usage : check_file_exist [-f <file> ou --file
<file>]

Missing argument: file

suspect:~ #
/usr/local/nagios/libexec/check_file_exist -V
check_file_exist 1.0

suspect:~ #
/usr/local/nagios/libexec/check_file_exist
--usage
Usage : check_file_exist [-f <file> ou --file
<file>]

suspect:~ #
/usr/local/nagios/libexec/check_file_exist -h
check_file_exist 1.0

Ce plugin Nagios est gratuit et libre de droits,
et vous pouvez l'utiliser à votre convenance. Il
est livré avec ABSOLUMENT AUCUNE GARANTIE.

Usage : check_file_exist [-f <file> ou --file
<file>]

 -?, --usage
   Print usage information
 -h, --help
   Print detailed help screen
 -V, --version
   Print version information
 --extra-opts=[section][@file]
   Read options from an ini file. See
http://nagiosplugins.org/extra-opts
   for usage and examples.
 -f, --file=STRING
   file to check
 -t, --timeout=INTEGER
   Seconds before plugin times out (default: 15)
 -v, --verbose
   Show details for command-line debugging (can
repeat up to 3 times)

suspect:~ #
/usr/local/nagios/libexec/check_file_exist -f
/var/log/messages
Verification fichier OK - messages exists

suspect:~ #
/usr/local/nagios/libexec/check_file_exist -f
/var/log/toto.txt
Verification fichier CRITICAL - toto.txt not
exists

gendarme:~ # /usr/local/nagios/libexec/check_nrpe
-n -H suspect_ou_IP_suspect -c check_file_exist
-a /var/log/messages
/usr/local/nagios/libexec/check_nrpe -n -H
suspect_ou_IP_suspect -c check_file_exist -a
/var/log/messages

gendarme:~ # /usr/local/nagios/libexec/check_nrpe
-n -H suspect_ou_IP_suspect -c check_file_exist
Usage : check_file_exist [-f <file> ou --file
<file>]
```

On constate que notre programme est assez simple avec un rendu propre et complet !! Le module **Nagios::Plugin** nous facilite la vie car nous n'avons plus à nous soucier de la gestion des arguments, il le fait pour nous ! Nous disposons d'une véritable documentation (un *man*). De plus, le formatage Nagios est respecté et le code est plus lisible. Ceci n'est qu'un exemple, mais le module peut nous permettre d'aller plus loin dans la documentation avec peu de code.

2.4.2.2. Plugin 2 - check_file_size

Le but de ce plugin est de vérifier la **taille** d'un **fichier** quelconque. Nagios devra nous notifier un warning, ou une alerte critique si le fichier dépasse les tailles que l'on passera en argument au plugin.

Avant de commencer, nous devons nous poser les questions suivantes :

- quel est le but de notre programme ?
- quels sont les arguments que notre plugin attend ?

En ce qui concerne le but, nous l'avons déjà, pour ce qui est des arguments, nous voulons fixer des seuils. Avec Nagios, les options utilisées pour les seuils de warnings et d'alertes critiques sont souvent respectivement **-w** et **-c**. Nous allons donc utiliser les mêmes. Les valeurs spécifiées à **-w** et **-c** devront être des entiers ou nombres réels. Elles représentent la taille en octets. Il nous faut une dernière option pour récupérer le fichier à tester. Voilà, nous pouvons commencer à coder !!

Chargeons notre module et créons notre constructeur en spécifiant la **version** de notre programme, son **nom** et une **licence**.

```
#!/usr/bin/perl
#===============================================
# Auteur : djibril
# Date   : 04/03/2011 09:16:24
# But    : Plugin vérifiant la taille d'un
fichier
#===============================================
use strict;
use warnings;

# Chargement du module
use Nagios::Plugin;
use File::Basename;

use vars qw/ $VERSION /;

# Version du plugin
$VERSION = '1.0';

my $LICENCE
  = "Ce plugin Nagios est gratuit et libre de
droits, et vous pouvez l'utiliser à votre
convenance."
  . ' Il est livré avec ABSOLUMENT AUCUNE
GARANTIE.';

my $plugin_nagios = Nagios::Plugin->new(
  shortname => 'Verification fichier',
  usage =>
    'Usage : %s [-f <file> ou --file <file>] [
-c|--critical=<threshold> ] [ -w|--
warning=<threshold> ]',
  version => $VERSION,
  license => $LICENCE,
);
```

Créons maintenant nos arguments.

```
# Définition de l'argument --file ou -f pour
récupérer le fichier
$plugin_nagios->add_arg(
  spec     => 'file|f=s',
  help     => 'File to check',     # Aide au sujet
de cette option
  required => 1,                   # Argument
obligatoire
);

# Définition de l'argument warning
$plugin_nagios->add_arg(
  spec     => 'warning|w=f',
# Nous acceptons des nombres réels
  help     => 'Exit with WARNING status if less
than BYTES bytes of file',
  label    => 'BYTES',
  required => 1,
);

# Définition de l'argument critical
$plugin_nagios->add_arg(
  spec     => 'critical|c=f',
  help     => 'Exit with CRITICAL status if less
than BYTES bytes of file',
  label    => 'BYTES',
  required => 1,
);
```

Nous avons créé trois options avec chacune un alias. **--file** pour **-f**, **--warning** pour **-w** et **--critical** pour **-c**. Remarquez que pour **-w** et **-c**, il y a une option **label => 'BYTES'**. Cette ligne permet au module d'afficher dans l'aide "*BYTES*" pour le type d'arguments à recevoir.

aide personnalisée

```
-w, --warning=BYTES
   Exit with WARNING status if less than BYTES
bytes of file
 -c, --critical=BYTES
   Exit with CRITICAL status if less than BYTES
bytes of file
```

Les options **-w** et **-c** sont obligatoires d'où les options **required => 1**. Activons maintenant l'analyse des arguments.

```
# Activer le parsing des options de ligne de
commande
$plugin_nagios->getopts;
```

Vérifions que le fichier à tester existe bien, si oui, calculons sa **taille**, si non envoyons une alerte critique.

```
my $fichier_a_verifier = $plugin_nagios->opts-
>file;
my $nom_fichier        = basename
$fichier_a_verifier;

if ( ! -e $fichier_a_verifier ) {
  $plugin_nagios->nagios_exit( CRITICAL,
"$nom_fichier not exists\n" );
}
```

Nous pouvons maintenant tester les seuils fixés par rapport à la taille du fichier pour afficher le message adéquat pour Nagios.

```
my $taille_fichier     = -s $fichier_a_verifier;

# Récupérons le code retour en fonction des
seuils
my $code_retour = $plugin_nagios-
>check_threshold(
  check    => $taille_fichier,
  warning  => $plugin_nagios->opts->warning,
  critical => $plugin_nagios->opts->critical,
);

$plugin_nagios->nagios_exit( $code_retour,
"Taille $nom_fichier ($taille_fichier)" );
```

La méthode **check_threshold** permet de spécifier une valeur à tester par rapport aux valeurs seuils pour retourner le code retour. Par exemple, si l'on souhaite qu'un fichier au-delà de 1000 octets émette (le plugin émet !) un warning et 2000 octets une alerte critique, il faudra les arguments -w 1000 -c 2000. Le plugin retournera le code retour adéquat (**0**, **1** ou **2**). Il nous suffit ensuite de renvoyer le message qui sera automatiquement adapté en fonction de ce code retour. Voilà le programme final !!

```
#!/usr/bin/perl
#=============================================
# Auteur : djibril
# Date   : 04/03/2011 09:16:24
# But    : Plugin vérifiant la taille d'un
fichier
#=============================================
use strict;
use warnings;

# Chargement du module
use Nagios::Plugin;
use File::Basename;

use vars qw/ $VERSION /;

# Version du plugin
$VERSION = '1.0';

my $LICENCE
  = "Ce plugin Nagios est gratuit et libre de
droits, et vous pouvez l'utiliser à votre
convenance."
  . ' Il est livré avec ABSOLUMENT AUCUNE
GARANTIE.';

my $plugin_nagios = Nagios::Plugin->new(
  shortname => 'Verification fichier',
  usage =>
    'Usage : %s [-f <file> ou --file <file>] [
-c|--critical=<threshold> ] [ -w|--
warning=<threshold> ]',
  version => $VERSION,
  license => $LICENCE,
);

# Définition de l'argument --file ou -f pour
récupérer le fichier
$plugin_nagios->add_arg(
  spec     => 'file|f=s',
  help     => 'File to check',    # Aide au sujet
de cette option
  required => 1,                  # Argument
obligatoire
);

# Définition de l'argument warning
$plugin_nagios->add_arg(
  spec     => 'warning|w=f',
# Nous acceptons des nombres réels
  help     => 'Exit with WARNING status if less
than BYTES bytes of file',
  label    => 'BYTES',
  required => 1,
);

# Définition de l'argument critical
$plugin_nagios->add_arg(
  spec     => 'critical|c=f',
  help     => 'Exit with CRITICAL status if less
than BYTES bytes of file',
  label    => 'BYTES',
  required => 1,
);

# Activer le parsing des options de ligne de
commande
$plugin_nagios->getopts;

my $fichier_a_verifier = $plugin_nagios->opts-
>file;
my $nom_fichier        = basename
$fichier_a_verifier;

if ( !-e $fichier_a_verifier ) {
  $plugin_nagios->nagios_exit( CRITICAL,
"$nom_fichier not exists\n" );
}

my $taille_fichier = -s $fichier_a_verifier;

# Récupérons le code retour en fonction des
seuils
my $code_retour = $plugin_nagios-
>check_threshold(
  check    => $taille_fichier,
  warning  => $plugin_nagios->opts->warning,
  critical => $plugin_nagios->opts->critical,
);

$plugin_nagios->nagios_exit( $code_retour,
"Taille $nom_fichier ($taille_fichier)" );

__END__
```

Configurons NRPE et Nagios pour la réception des arguments.

/usr/local/nagios/nrpe.cfg

```
vi /usr/local/nagios/nrpe.cfg

command[check_file_size]=/usr/local/nagios/libexe
c/check_file_size -f $ARG1$ -w $ARG2$ -c $ARG3$
```

Une fois le fichier nrpe.cfg modifié, nous pouvons faire un test en ligne de commande depuis le serveur **"gendarme"**.

```
/usr/local/nagios/libexec/check_nrpe -n -H
suspect_ou_IP_suspect -c check_file_size -a
/var/log/messages 154524 21565658
Verification fichier WARNING - Taille messages
(2589788)
```

Le résultat montre que notre fichier a une taille comprise entre 154524 et 21565658, d'où le warning.
Configurons Nagios sur le serveur de supervision pour permettre à Nagios de faire les vérifications.

```
define service{
    use                     generic-service
    host_name               suspect_ou_IP_suspect
    service_description     Check size
    check_command           check_nrpe!
check_file_size!/var/log/messages 154524 21565658
}
```

Voilà, tout fonctionne parfaitement !! Vous pouvez encore améliorer ce plugin afin de permettre saisir en argument une option qui donnera la possibilité de définir les seuils en Ko, Mo, Go, etc. Cela peut être un bon exercice.

2.4.2.3. Plugin 3 - check_process

Le but de ce plugin est de vérifier qu'un processus quelconque tourne sur une machine Linux via l'argument (**-n** ou **--proc_name**). Si le processus n'est pas trouvé, Nagios émettra une alerte critique sinon tout est OK.

Nous utiliserons le module Perl Proc::ProcessTable (Lien 105) qui permet de lister facilement les processus sous Linux. Grâce à ce module, nous pouvons vérifier que le processus cherché existe et tourne. Voici notre plugin :

```
#!/usr/bin/perl
#=============================================
# Auteur : Djibril
# But    : Plugin Nagios permettant de vérifier
l'existence d'un
#          processus via le nom
# Date   : 05/03/2011 12:38:44
#=============================================

use warnings;
use strict;
use Nagios::Plugin;
use Proc::ProcessTable;

use vars qw/ $VERSION /;

# Version du plugin
$VERSION = '1.0';

my $LICENCE
  = "Ce plugin Nagios est gratuit et libre de
droits, et vous pouvez l'utiliser à votre
convenance."
  . ' Il est livré avec ABSOLUMENT AUCUNE
GARANTIE.';

my $plugin_nagios = Nagios::Plugin->new(
  shortname => 'Check process',
  usage     => 'Usage: %s [ -proc_name|-
n=<Process name> ]',
  version   => $VERSION,
  license   => $LICENCE,
);

# Définition des options de ligne de commande
# 1- Récupération du nom du processus
$plugin_nagios->add_arg(
  spec     => 'proc_name|n=s',
  help     => 'Process name to find',
  required => 1,
);

# Parser les options
$plugin_nagios->getopts;
my $proc_name = $plugin_nagios->opts->proc_name;

# Recherche du processus
my $process_table = new Proc::ProcessTable;
my $FORMAT        = "%-6s %-10s %-8s %-24s %s\n";
foreach my $proc ( @{ $process_table->table } ) {
  if ( $proc->fname eq $proc_name ) {
    my $message_OK = "Process $proc_name (" .
$proc->pid . ') state (' . $proc->state . ')';

    # Option verbose
    if ( $plugin_nagios->opts->verbose ) {
      printf( $FORMAT, 'PID', 'TTY', 'STAT',
'START', 'COMMAND' );
      printf( $FORMAT,
        $proc->pid, $proc->ttydev, $proc->state,
scalar( localtime( $proc->start ) ),
        $proc->cmndline );
    }
    $plugin_nagios->nagios_exit( OK,
$message_OK );
  }
}
$plugin_nagios->nagios_exit( CRITICAL,
"$proc_name not found\n" );

__END__
```

Explication du programme :

Ce plugin liste tous les processus Linux tournant sur la machine grâce à la méthode **table** du module **Proc::ProcessTable**. Pour chaque processus, nous vérifions si le nom du processus est égal à celui que nous cherchons (méthode **fname**). Si le processus et trouvé, le plugin sort avec le bon code de retour et un message spécifiant le **nom**, l'**état** et le **pid** du processus. L'utilisation de l'option verbose en ligne de commande donne plus d'informations sur le processus trouvé. Exemple :

```
suspect:~ #
/usr/local/nagios/libexec/check_process_ah -n
mysqld -v
PID    TTY        STAT     START
COMMAND
3646               sleep    Wed Apr  7 09:30:25
2010 /usr/sbin/mysqld --basedir=/usr
--datadir=/var/lib/mysql --user=mysql --pid-
file=/var/lib/mysql/mysqld.pid --skip-external-
locking --port=3306
--socket=/var/lib/mysql/mysql.sock
Check process OK - Process mysqld (3646) state
(sleep)
```

Configurons NRPE et Nagios pour la réception des arguments.

/usr/local/nagios/nrpe.cfg

```
vi /usr/local/nagios/nrpe.cfg

command[check_process]=/usr/local/nagios/libexec/
check_process -n $ARG1$
```

Configurons Nagios sur le serveur de supervision pour permettre à Nagios de faire la vérification.

```
define service{
    use                     generic-service
    host_name               suspect_ou_IP_suspect
    service_description     Check Process Mysqld
    check_command           check_nrpe!
check_process!mysqld
}
```

La configuration ci-dessus permettra à Nagios de vérifier que le service mysqld tourne sur le serveur **"suspect"**.

Voilà !!

3. Références utilisées

- Installation et configuration de Nagios pour débutants : Lien 106
- Documentation officielle de Nagios : Lien 107
- Modules Perl : Proc::ProcessTable (Lien 105) et Nagios::Plugin (Lien 102)
- API pour les plugins Nagios : Lien 108
- Projet Plugins Nagios : Lien 109
- Les plugins de Nagios depuis son site officiel : Lien 110
- NagiosExchange qui propose des centaines de plugins gratuits : Lien 111
- Le site officiel des grandes lignes du développement des plugins Nagios : Lien 112

4. Conclusion

J'espère que ce tutoriel vous a permis de comprendre et de prendre conscience que le choix du langage Perl pour concevoir un plugin Nagios est un bon choix ! Nous avons vu comment créer des programmes respectant parfaitement les **normes** Nagios contrairement à beaucoup de plugins gratuits que l'on peut trouver sur la Toile. De plus, Perl est bien adapté pour les réseaux, la gestion de fichiers... N'hésitez donc pas à faire vos plugins à partir de ce tutoriel.

Retrouvez l'article de djibril en ligne : Lien 113

Perl et la programmation orientée objet - De la base à la modernité

Le but de cet article est de vous exposer la façon la plus simple et la plus classique de faire de la Programmation Orientée Objet en Perl, puis de vous montrer les dernières recommandations de la communauté Perl.

1. Brève introduction de Perl et de la Programmation Orientée Objet

La **P**rogrammation **O**rientée **O**bjet (qu'on notera **POO**) est un concept possédant de nombreuses vertus universellement reconnues à ce jour. C'est une méthode de programmation qui permet d'améliorer le développement et la maintenance d'applications, de logiciels avec un gain de temps non négligeable. Il est important de garder à l'esprit qu'elle ne renie pas la programmation structurée (ou procédurale) puisqu'elle est fondée sur elle. L'approche classique pour introduire la POO dans un langage existant est de passer par une encapsulation des fonctionnalités existantes. Le langage C++ s'est construit sur le C et a apporté la POO. Le langage JAVA est fondé sur la syntaxe du C++.
Perl a également suivi le pas en proposant des extensions pour donner la possibilité à ses fans de pouvoir utiliser ce paradigme de programmation. Néanmoins, Perl étant permissif, il n'est pas aussi strict que des langages "pur objet". Mais c'est normal, car la philosophie de Perl est maintenue, *"there is more than one way to do it"* (**TIMTOWTDI**).

Dans les sections **A** et **B** nous allons vous expliquer comment fonctionne la POO en Perl traditionnellement, néanmoins en Perl moderne, certains modules facilitent grandement et rendent bien plus élégante la POO, nous vous les présenterons en section **C**.

1.1. Avantages

Il est plus facile et rapide de modifier, de faire évoluer, de maintenir du code issu d'un logiciel, d'une application de moyenne ou de grande envergure avec la POO.
L'architecture du code peut permettre à d'autres applications de réutiliser des composants, des classes...
D'autres développeurs peuvent facilement utiliser vos programmes. C'est le cas des modules du CPAN que l'on a l'habitude d'utiliser : ils sont tous écrits en POO ! Je tiens à tout de même préciser que les modules simples sont aussi réutilisables. La clarté d'un module n'est pas liée au paradigme de programmation.

1.2. Inconvénients

La philosophie de Perl peut être un inconvénient à la POO car il existe des dizaines de façons différentes d'écrire un programme, des modules...
Sachez que la POO dégrade en général les performances et qu'il est important de savoir si ce paradigme de programmation est adapté à votre problématique. La POO nécessite sans doute une plus grande réflexion quant à la définition de l'architecture du programme, avant de passer à la programmation (ce qui peut être un avantage). Dans tous les cas, comme le disait **Damian Conway**, *"Utilisez la POO pour ses avantages et malgré ses inconvénients et pas simplement parce que c'est le gros marteau familier et confortable qui figure dans votre boite à outils"*.

2. Les bases de la POO en Perl

2.1. Les classes et les objets

Lorsque l'on débute en POO, les notions de **classe** et d'**objet** peuvent nous paraître abstraites. Une mauvaise compréhension peut être fatale pour la suite de cet article.

Voici une explication de *Sylvain Lhuiller* que je trouve simple et concise.

- **Explication de la POO en Perl par Sylvain Lhuiller**

La programmation orientée objet est un type de programmation qui se concentre principalement sur les données. La question qui se pose en POO est *« quelles sont les données du problème ? »* par opposition à la programmation procédurale par exemple, qui pose la question *« quelles sont les fonctions/actions à faire ? »*. En POO, on parle ainsi d'objets, auxquels on peut affecter des **variables/attributs (propriétés)** et des **fonctions/actions (méthodes)**.

On parle de **« classe »**, qui est une manière de représenter des données et comporte des traitements : une classe **« Chaussure »** décrit, par exemple, les caractéristiques d'une chaussure. Elle contient un champ décrivant la *pointure*, la *couleur*, la *matière*, etc. Une telle classe comporte de plus des traitements sur ces données ; ces traitements sont appelés **« méthodes »**. Grossièrement une méthode est une fonction appliquée à un objet. Exemples de méthode : monter, coudre, ressemeler...

Une fois définie une telle classe, il est possible d'en construire des instances : une instance de classe est dite être un objet de cette classe. Dans notre exemple, il s'agirait d'une chaussure dont la pointure, la couleur et la matière sont renseignées.

2.1.1. Définition d'une classe

Commençons les choses sérieuses en parlant Perl! Vous verrez qu'il est simple de parler Perl avec l'accent orienté objet !

Définir une classe est très simple car ce n'est rien d'autre qu'un package, un module en Perl. Un objet n'est autre chose qu'une référence à un scalaire, un hachage, un tableau (voire une fonction, un typeglob...) qui est liée à cette classe.

Voici notre classe **"Personne"** :

Personne.pm
```
package Personne; # Nom du package, de notre classe
use warnings;     # Avertissement des messages d'erreurs
use strict;       # Vérification des déclarations
use Carp;         # Utile pour émettre certains avertissements
# ...
# ...

1;                # Important, à ne pas oublier
__END__           # Le compilateur ne lira pas les lignes après elle
```

Voilà! notre classe est créée, simple non !! Il suffit de créer un package du nom de la classe dans un fichier .pm du même nom.

Notre script principal appelle la classe via un **use** habituel.

ScriptPrincipal.pl
```
use Personne;
```

2.1.2. Définition d'un constructeur

Pour pouvoir créer un objet et instancier la classe **"Personne"**, on doit définir un **constructeur** dans la classe, c'est-à-dire le fichier *Personne.pm* (sachez qu'un script ".pm" peut avoir plusieurs classes, mais dans cet article, on travaillera avec une classe par fichier ".pm").
Créer un constructeur Perl revient à créer une fonction spéciale nommée **"new"**. Le constructeur peut être nommé **"create"**, **"forge"** ou tout autre nom. Et il peut exister plusieurs constructeurs. Perl n'impose pas grand-chose sur le constructeur comme dans les autres langages OO. Néanmoins, c'est le nom standard utilisé en Perl POO. Dans un souci de maintenance et de clarté, pourquoi déroger à la règle !! Sachez seulement que ce n'est pas un opérateur comme dans certains langages POO.

Personne.pm - constructeur
```
sub new {
  # ...
}
```

L'appel du constructeur dans le script principal se fait de la façon suivante :

ScriptPrincipal.pl
```
my $Personne = Personne->new();
```

On reviendra plus tard sur cet appel.

Notre classe **"Personne"** est caractérisée par son nom, son prénom, son âge, son sexe et son nombre d'enfants. Ces informations sont transmises par l'utilisateur de notre classe (le script principal) au constructeur. Ce dernier s'attend donc à 5 arguments. Mais en fait non ! Ce sera 6 car le premier argument correspond toujours au nom de la classe. Mais l'utilisateur de la classe n'a pas à se soucier de cela, il ne passe que 5 arguments au constructeur. Créons 5 champs (attributs), attributs pour cette classe (*nom, prénom, âge, sexe, nombre d'enfants*).

Personne.pm - constructeur
```
sub new {
  my ( $classe, $nom, $prenom, $age, $sexe, $nombre_enfant ) = @_;
  # ...
}
```

Tout constructeur doit déterminer si son premier argument est le nom de la classe ou une référence :

Personne.pm - constructeur : vérification de la classe
```
$classe = ref($classe) || $classe;
```

Si **$classe** est une référence (ce qui signifie que c'est une instance d'une classe, on en parlera plus loin dans l'article), on prend ref($classe) (c'est-à-dire que le constructeur a été appelé à partir d'un objet *$djibril->new();*), sinon, $classe est la chaîne contenant la classe (c'est-à-dire que le constructeur a été appelé à partir du nom de la classe : *Personne->new();*). *ref* retourne le nom de la classe lorsque la référence passée en paramètre est une référence à un objet (et non une référence simple, non blessed).

Vous pouvez d'ores et déjà tester pour vous amuser.

Personne.pm - constructeur : vérification de la classe
```
$classe = ref($classe) || $classe;
print $classe;
```

Script principal
```
#!/usr/bin/perl
use warnings;
use strict;
use Carp;

use Personne;

my $Personne = Personne->new();
```

Cela a pour effet d'afficher la chaîne *Personne*.

Créons maintenant une référence anonyme à un hachage vide qui stockera les attributs de notre classe et deviendra l'objet de la classe. Nous la nommons **$this**.

Personne.pm - constructeur : futur objet
```
my $this = {};
```

Si vous avez un souci de compréhension des références Perl, notre FAQ Perl est votre amie : Lien 114.

Nous avons choisi de nommer notre variable « $this » par analogie aux autres langages OO, mais ce n'est pas obligatoire. D'ailleurs, si vous consultez le code source de certains modules Perl du CPAN, vous verrez qu'elles sont souvent nommées **$self**.

Lions notre variable $this à notre classe :

Personne.pm - constructeur : liaison de la référence à notre classe
```
bless($this, $classe);
```

A ce stade, notre objet est créé et lié à la classe **"Personne"**. Si vous l'affichez (*print $this*), vous obtiendrez ceci ***Personne=HASH(0x235bb0)*** alors qu'avant la *bénédiction* (bless), on obtenait ***HASH(0x235bb0)***. L'adresse mémoire correspondant au hachage est liée à la classe.

Vous ne dormez pas encore j'espère !!

Stockons maintenant les attributs de notre classe que l'utilisateur passera via le constructeur.

Personne.pm - constructeur : arguments
```
$this->{_NOM}           = $nom;
$this->{_PRENOM}        = $prenom;
$this->{AGE}            = $age;
$this->{_SEXE}          = $sexe;
$this->{NOMBRE_ENFANT}  = $nombre_enfant;
```

Voilà, fini !! Plus qu'à retourner l'objet à l'utilisateur !

Personne.pm - constructeur : retourner l'objet
```
return $this;
```

Vous aurez remarqué que les attributs d'un constructeur (noms des clés) sont en **majuscules**. De plus, certaines clés sont **précédées d'un souligné "_"**, cela signifie que les valeurs des clés ne devraient pas être modifiables par l'utilisateur. Ce ne sont que des conventions, mais je vous les recommande.
Dans notre exemple, une personne ne peut pas changer de nom, de prénom et de sexe !!

Voilà à quoi ressemble notre premier constructeur !

Personne.pm - constructeur
```
sub new {
  my ( $classe, $nom, $prenom, $age, $sexe,
$nombre_enfant ) = @_;

  # Vérifions la classe
  $classe = ref($classe) || $classe;

  # Création de la référence anonyme de hachage
vide (futur objet)
  my $this = {};

  # Liaison de l'objet à la classe
  bless $this, $classe;

  $this->{_NOM}           = $nom;
  $this->{_PRENOM}        = $prenom;
  $this->{AGE}            = $age;
  $this->{_SEXE}          = $sexe;
  $this->{NOMBRE_ENFANT}  = $nombre_enfant;

  return $this;
}
```

2.1.3. Les objets

Avec ce peu de code, on peut déjà instancier notre classe en créant des objets depuis notre script principal.

Script principal - Objet
```
#!/usr/bin/perl
use warnings;
use strict;
use Carp;

use Personne;

my $Objet_Personne1 = Personne->new('Dupont',
'Jean', 45, 'M', 3);
```

Nous venons de créer notre objet, on pourrait en créer plusieurs.
Remarquez bien la notation : la classe est suivie d'une flèche qui pointe sur le constructeur **"new"** auquel on passe des arguments. Il existe une autre notation correcte, mais pouvant porter à confusion.

Script principal - Objet
```
my $Objet_Personne1 = new Personne('Dupont',
'Jean', 45, 'M', 3);
```

Cela donne l'impression que **new** est un opérateur spécifique de Perl et que l'on appelle une méthode **Personne** à laquelle on passe des arguments. Ce qui est bien sûr faux ! Donc gardez en tête la notation recommandée avec flèche.

Retrouvez la suite de l'article de djibril en ligne : Lien 115

Liens

Lien 01 : http://www.developpez.com/actu/28861
Lien 02 : http://www.developpez.com/actu/16080
Lien 03 : http://nighthacks.com/roller/jag/entry/next_step_on_the_road
Lien 04 : http://www.developpez.net/forums/d1058102/club-professionnels-informatique/actualites/james-gosling-pere-java-rejoint-google/
Lien 05 : http://eclipse.developpez.com/actu/25420/Google-offre-une-partie-des-outils-d-Instantiations-a-la-communaute-open-source-ils-deviennent-des-projets-Eclipse/
Lien 06 : http://www.eclipsecon.org/2011/program/?programdate=2011-03-22
Lien 07 : http://www.developpez.net/forums/d1057986/environnements-developpement/eclipse/eclipse-3-7-sortira-22-juin-prochain/
Lien 08 : http://www.symantec.com/connect/blogs/android-threat-tackles-piracy-using-austere-justice-measures
Lien 09 : http://www.developpez.net/forums/d1059620/java/general-java/java-mobiles/android/application-android-piratee-lutte-contre-piratage/
Lien 10 : http://www.developpez.net/forums/d1056923/java/general-java/java-mobiles/android/android-permet-paiement-vente-applications/
Lien 11 : http://sberfini.developpez.com/tutoriaux/android/bluetooth/
Lien 12 : http://sberfini.developpez.com/tutoriaux/android/nfc
Lien 13 : http://sberfini.developpez.com/tutoriaux/android/nfc/bluetooth/
Lien 14 : http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html
Lien 15 : http://developer.android.com/reference/android/content/ContentValues.html
Lien 16 : http://developer.android.com/reference/android/content/ContentResolver.html
Lien 17 : http://developer.android.com/reference/android/database/Cursor.html
Lien18 : http://a-renouard.developpez.com/tutoriels/android/sqlite/
Lien 19 : http://android.developpez.com/cours/specialisation-ressources/
Lien 20 : http://julien-pauli.developpez.com/tutoriels/php/sapis/
Lien 21 : http://www.php.net/outcontrol
Lien 22 : http://www.php.net/output-add-rewrite-var
Lien 23 : http://julien-pauli.developpez.com/tutoriels/php/sapis/
Lien 24 : http://julien-pauli.developpez.com/tutoriels/php/internals/presentation/
Lien 25 : http://julien-pauli.developpez.com/tutoriels/php/ob/
Lien26 : http://fr.wikipedia.org/wiki/Base_%28arithmétique%29
Lien 27 : http://www.php.net/operators.bitwise
Lien 28 : http://fr.wikipedia.org/wiki/Algèbre_de_Boole_%28logique%29
Lien29 : http://php.net/errorfunc.constants
Lien 30 : http://www.libmng.com/pub/png/spec/1.2/PNG-Structure.html
Lien 31 : http://www.php.net/pack
Lien 32 : http://dvsoft.developpez.com/Articles/CRC/
Lien 33 : http://emmanuel-delahaye.developpez.com/tutoriels/c/operateurs-bit-bit-c/
Lien 34 : http://julien-pauli.developpez.com/tutoriels/php/bool-op/
Lien 35 : http://cssglobe.com/post/9435/pure-css3-post-tags
Lien 36 : http://cssglobe.developpez.com/
Lien 37 : http://debray-jerome.developpez.com/demos/posts.html
Lien 38 : http://www.dinnermint.org/blog/css/creating-triangles-in-css
Lien 39 : http://cssglobe.developpez.com/tutoriels/css/tags-pour-post-en-css3/
Lien 40 : http://debray-jerome.developpez.com/demos/transformations.html
Lien 41 : http://debray-jerome.developpez.com/articles/les-transformations-en-css3/
Lien 42 : http://debray-jerome.developpez.com/articles/geometrie-avec-css/
Lien 43 : http://jquery.mathieurobin.com/slideshow/
Lien 44 : http://api.jquery.com/delegate/
Lien 45 : http://www.flickr.com/
Lien 46 : http://www.flickr.com/photos/davidden/244741330
Lien 47 : http://www.flickr.com/photos/shayhaas/426375654
Lien 48 : http://www.flickr.com/photos/83085326@N00/4909713714
Lien 49 : http://www.flickr.com/photos/divemasterking2000/4767003134
Lien 50 : http://mathieu-robin.developpez.com/tutoriels/javascript/creer-plugin-slideshow-pour-jquery/
Lien 51 : http://www.sohtanaka.com/web-design/inline-modal-window-w-css-and-jquery/
Lien 52 : http://sohtanaka.developpez.com/
Lien 53 : http://fancybox.net/
Lien 54 : http://www.no-margin-for-errors.com/projects/prettyphoto-jquery-lightbox-clone/
Lien 55 : http://sohtanaka.developpez.com/tutoriels/javascript/creez-fenetre-modale-avec-css-et-jquery/fichiers
Lien 56 : http://www.jquery.com/
Lien 57 : http://jquery.com/
Lien 58 : http://sohtanaka.developpez.com/tutoriels/javascript/creez-fenetre-modale-avec-css-et-jquery/
Lien 59 : http://erwy.developpez.com/tutoriels/xml/xpath-langage-selection-xml/
Lien 60 : http://erwy.developpez.com/tutoriels/xml/xpath-fonctionnements-predicats/
Lien 61 : http://erwy.developpez.com/tutoriels/xml/xpath-fonctionnements-predicats/#L4
Lien 62 : http://erwy.developpez.com/tutoriels/xml/xpath-fonctionnements-predicats/#L3-C
Lien 63 : http://erwy.developpez.com/tutoriels/xml/xpath-liste-fonctions/
Lien 64 : http://www.developpez.com/actu/13013
Lien 65 : http://herbsutter.com/2011/03/25/we-have-fdis-trip-report-march-2011-c-standards-meeting/
Lien 66 : http://www.developpez.net/forums/d891380/c-cpp/cpp/cpp0x-draft-final-ete-vote/
Lien 67 : http://blog.qt.nokia.com/2011/03/24/qt-creator-2-2-beta-released/
Lien 68 : http://labs.qt.nokia.com/2011/03/24/qt-creator-2-2-beta-release/
Lien 69 : http://www.developpez.net/forums/d1056554/c-cpp/bibliotheques/qt/edi/qt-creator/qt-creator-2-2-beta/
Lien 70 : http://doc.trolltech.com/extras/qt43-class-chart.pdf
Lien 71 : http://qt.developpez.com/doc/arbre/42/QBoxLayout/
Lien 72 : http://qt.developpez.com/doc/4.7/qboxlayout
Lien 73 : http://www.developpez.net/forums/d941430/c-cpp/bibliotheques/qt/documentation-qt-4-7-francais-arbre-classes/
Lien 74 : http://developer.qt.nokia.com/wiki/PySideDownloads/
Lien 75 : http://www.pyside.org/2011/03/pyside-python-for-qt-1-0-released/
Lien 76 : http://www.developpez.net/forums/d1014139/autres-langages/python-zope/gui/pyside-pyqt/sortie-pyside-1-0-0-rc1/

Lien 77 : http://www.glc-lib.net/download.php
Lien 78 : http://pkgs.org/
Lien 79 : http://www.glc-lib.net/examples.php
Lien 80 : http://www.glc-lib.net/help.php
Lien 81 : http://www.glc-lib.net/doc/index.html
Lien 82 : http://www.developpez.net/forums/f1553/c-cpp/bibliotheques/qt/outils/bibliotheques/glc_lib/
Lien 83 : http://www.glc-lib.net/forum/
Lien 84 : http://laurent-ribon-glc-lib.developpez.com/tutoriels/qt/presentation-glc-lib/fichiers/glc_BasicViewer.zip
Lien 85 : http://laurent-ribon-glc-lib.developpez.com/tutoriels/qt/presentation-glc-lib/
Lien 86 : http://securite.developpez.com/livres/
Lien 87 : http://www.youtube.com/watch?v=lnedOWfPKT0
Lien 88 : http://www.developpez.net/forums/d1050128/systemes/securite/premier-virus-pc-25-ans/
Lien 89 : http://fr.wikipedia.org/wiki/Hameçonnage#cite_note-JORF-0
Lien 90 : http://fr.wikipedia.org/wiki/Attaque_par_déni_de_service
Lien 91 : http://fr.wikipedia.org/wiki/Conficker
Lien 92 : http://securite.developpez.com/actu/29301/Le-Ministere-de-l-Economie-et-des-Finances-touche-par-une-attaque-sophistiquee-la-premiere-contre-l-Etat-francais-de-cette-ampleur/
Lien 93 : http://securite.developpez.com/
Lien 94 : http://securite.developpez.com/livres/?page=sommaire
Lien 95 : http://www.ed-diamond.com/produit.php?ref=misc41&id_rubrique=8&caracteristique=1-2-&caracdisp=2-9-
Lien 96 : http://thpierre.developpez.com/articles/pieges-internet/
Lien 97 : http://www.securiteinfo.com/divers/documentation_securite_informatique.shtml
Lien 98 : http://www.hoaxkiller.fr/questce/generalites.htm
Lien 99 : http://simon-sayce.developpez.com/tutoriels/securite/malwares-pour-nuls/
Lien 100 : http://djibril.developpez.com/tutoriels/linux/nagios-pour-debutant/
Lien 101 : http://www.nagios.org/documentation
Lien 102 : http://search.cpan.org/search?query=Nagios%3A%3APlugin&mode=all
Lien 103 : http://nagiosplug.sourceforge.net/developer-guidelines.html
Lien 104 : http://djibril.developpez.com/tutoriels/perl/installation-modules/
Lien 105 : http://search.cpan.org/search/?query=Proc%3A%3AProcessTable&mode=all
Lien 106 : http://djibril.developpez.com/tutoriels/linux/nagios-pour-debutant/
Lien 107 : http://www.nagios.org/documentation
Lien 108 : http://doc.monitoring-fr.org/3_0/html/development-pluginapi.html
Lien 109 : http://nagiosplug.sourceforge.net/
Lien 110 : http://www.nagios.org/download/plugins/
Lien 111 : http://exchange.nagios.org/
Lien 112 : http://nagiosplug.sourceforge.net/developer-guidelines.html
Lien 113 : http://djibril.developpez.com/tutoriels/perl/ecrire-facilement-plugin-nagios-perl/
Lien 114 : http://perl.developpez.com/faq/perl/?page=sectionC4
Lien 115 : http://djibril.developpez.com/tutoriels/perl/poo/