



Develloppez

Le Mag

Édition de Décembre - Janvier 2010/2011.
Numéro 31.

Magazine en ligne gratuit.

Diffusion de copies conformes à l'original autorisée.

Réalisation : Alexandre Pottiez

Rédaction : la rédaction de Develloppez

Contact : magazine@redaction-developpez.com

Sommaire

Java	Page 2
Android	Page 4
Spring	Page 9
PHP	Page 10
Zend Framework	Page 17
(X)HTML/CSS	Page 21
Apache	Page 27
Flash/Flex	Page 32
JavaScript	Page 34
C/C++/GTK+	Page 40
Qt	Page 41
Windows Phone	Page 44
Mac	Page 54
iOS	Page 55
Conception	Page 57
Liens	Page 69

Article Zend Framework



Mettez en œuvre les captchas avec Zend Framework

Cet article va présenter la problématique d'utilisation d'un captcha, puis plusieurs solutions de mise en oeuvre dans un environnement Php/Zend.

par **Pierre Schwartz**
Page 17

Article Windows Phone



Introduction au développement d'applications Windows Phone 7

Cet article constitue une introduction au développement d'applications pour Windows Phone 7. Il est le premier d'une série traitant de divers aspects de programmation sous ce nouvel environnement.

par **Nico-pyright(c)**
Page 44

Éditorial

Ce mois-ci, la rédaction vous offre un nombre record de rubriques dans son magazine et, pour assouvir toujours plus votre soif de connaissance, il n'y en a pas moins de quatre nouvelles.

Profitez-en bien !

La rédaction



Enfin des JSR pour Java 7 & 8

Les JSR sont des documents primordiaux du **Java Community Process**, puisqu'il s'agit des documents de travail qui seront soumis aux votes de la part des membres du-dit **JCP**.

Longtemps attendue, la **JSR de Java 7** est enfin là, et elle n'est pas seule, puisqu'on y retrouve en tout quatre nouvelles JSRs :

- JSR 334 : Small Enhancements to the Java™ Programming Language (Le projet Coin pour **Java 7**) : [Lien1](#) ;
- JSR 335 : Lambda Expressions for the Java™ Programming Language (le projet Lambda pour **Java 8**) : [Lien2](#) ;
- JSR 336 : Java™ SE 7 Release Contents : [Lien3](#) ;
- JSR 337 : Java™ SE 8 Release Contents : [Lien4](#).

Java SE 7

Comme convenu, **Java SE 7** se limitera à trois principales JSRs :

- JSR 203 : More New I/O APIs for the Java Platform ("NIO.2") : [Lien5](#)
 - **API FileSystem** permettant un accès complet au système de fichiers (en remplacement de la classe `File` très limitée) (Lire : NIO.2 : Accès au système de fichiers dans Java 7 ([Lien6](#))) ;
 - API d'E/S asynchrone pour les fichiers et les sockets ;
 - Finalisation des fonctionnalités de la *JSR 51* sur les `SocketChannel` (binding, configuration et multicast).
- JSR 292 : Supporting Dynamically Typed Languages on the Java Platform : [Lien7](#)
 - Support de l'instruction **invokedynamic** afin de supporter des langages dynamiques directement au niveau du bytecode ;
 - API **java.dyn** permettant d'utiliser cela en **Java (InvokeDynamic et MethodHandle)**.
- JSR 334 : Small Enhancements to the Java Programming Language (OpenJDK Project Coin) : [Lien1](#)
 - **Switch** sur les chaînes de caractères (`String`) ;
 - Amélioration des valeurs numériques (valeurs en binaire et underscores) ;
 - Multi-catch et rethrow évolué (Lire : Gestion des exceptions améliorée (multi-catch et rethrow) ([Lien8](#))) ;
 - Amélioration du "Type Inference" des **Generics**, et syntaxe en losange (Diamond syntax) ;
 - Gestion automatique des ressources

(ARM aka **try-with-resources**) (Lire : Java 7 et les try-with-resources (ARM block) ([Lien9](#))) ;

- Simplified Varargs Method Invocation.

Lire : Projet Coin : Les modifications du langage pour Java 7 (note : le support syntaxique des collections a été reporté pour Java 8) : [Lien10](#).

Tout en incluant des changements de maintenance des JSRs existantes, et un certain nombre de "petites" modifications :

- Thread-safe concurrent class loaders ;
- Unicode 6.0 ;
- Enhanced locale support (IETF BCP 47 and UTR 35) ;
- TLS 1.2 ;
- Elliptic-curve cryptography ;
- JDBC 4.1 ;
- Standardisation des APIs de Java 6u10 (Translucent and shaped windows, Heavyweight/lightweight component mixing, Swing Nimbus look-and-feel, Swing JLayer component).

Java SE 8

Côté **Java 8** c'est déjà plus intéressant car on découvre pour la première fois la proposition des JSRs qui le composera :

- JSR 308 : Annotations on Java Types : [Lien11](#)
 - Généralisation de l'utilisation des annotations sur tous les types, et non plus uniquement sur les classes/méthodes/attributs/variables.
- JSR 310 : Date and Time API : [Lien12](#)
 - Une nouvelle API en remplacement de l'obsolète classe `Date`, qui intègrera des notions totalement absentes de l'API actuelle (instant, durée, intervalles, etc).
- JSR ??? : More Small Enhancements to the Java Programming Language (OpenJDK Project Coin)
 - On peut penser que le support syntaxique des collections fera partie du lot ;
 - Autre chose ? (la conférence de DevOxx pourrait nous en apprendre plus).
- JSR 335 : Lambda Expressions for the Java Programming Language (OpenJDK Project Lambda) : [Lien2](#)
 - Les expressions Lambda (équivalent d'une méthode anonyme) ;
 - Conversion vers les types SAM ;
 - Method references ;
 - Exception transparency ;
 - Public Defenders Methods (Lire : Evolution des interfaces avec les "public defenders methods" ([Lien13](#))).

Lire : Petit état des lieux du projet Lambda... ([Lien14](#))

- JSR ??? : Java Platform Module System

Dommmage collatéral

On peut noter également le report indéterminé de trois JSRs :

- JSR 260 : Javadoc Tag Technology Update : [Lien15](#)
 - Rien d'étonnant car cette vieille JSR avait déjà été reportée. A mon avis elle sera purement abandonné.
- JSR 295 : Beans Binding : [Lien16](#)
 - Il est trop tôt pour cela, car c'est typiquement le genre d'API qui gagnera en qualité avec l'utilisation des expressions Lambda.
De plus pendant les conférences de Devoxx il a été question d'intégrer à **Java** un vrai système de **property**. Si binding il y a, il devrait être basé là-dessus !
Bref il est préférable d'attendre le JDK 8

au minimum !

- JSR 296 : Swing Application Framework : [Lien17](#)
 - Aucune idée... et vu l'évolution lente du JDK et de Swing, je ne sais même pas si ce serait une bonne idée.
A titre personnel, et au risque d'en faire hurler certains, j'opterais pour une refonte totale de la couche UI.

Enfin une date ?

En pleine conférence **Devoxx**, il semblerait que les dates commencent à se confirmer.

Le **JDK7** devrait être disponible en juillet 2010, tandis que le **JDK8** arrivera fin 2012 (en même temps que la fin du monde selon les Mayas - il y a peut-être une relation de cause à effet !)

Retrouvez ce billet sur le blog de Frédéric Martini : [Lien18](#)

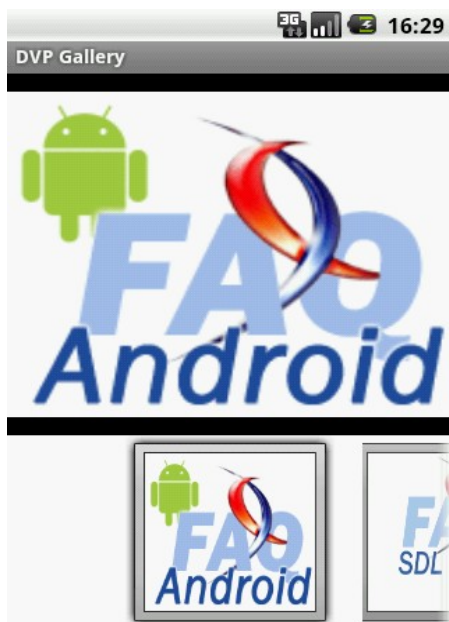
Création d'une galerie connectée

Cet article va vous permettre de comprendre comment fonctionne le composant 'Gallery', mais également comment réaliser des appels serveurs afin de télécharger des images, puis les afficher.

1. Présentation

Sous Android, le composant pour afficher une liste d'images est la Gallery. On y retrouve le même fonctionnement qu'une Listview classique ([Lien19](#)) : un Adapter est utilisé pour créer chacun des items de la liste. L'autre partie de ce tutoriel va vous montrer comment télécharger des ressources depuis Internet et s'en servir.

Le but étant d'arriver à un résultat proche de celui-ci :



2. Création du Layout

Notre écran se veut très simple afin de cibler les deux ou trois points de ce tutoriel. Dans un LinearLayout, on va venir y placer un composant qui affichera l'image en plein écran (ImageView), ainsi qu'un composant qui affichera la liste des images (Gallery).

Layout main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
    android:orientation="vertical"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent">

    <ImageView android:id="@+id/imageview"
        android:layout_width="fill_parent"
        " android:src="@drawable/no_photo"
        android:layout_height="wrap_conte
```

```
nt" android:layout_weight="0.2"></ImageView>
    <Gallery android:id="@+id/gallery"
        android:layout_width="fill_parent"
            android:layout_height="wrap_conte
nt" android:background="@color/blanc"></Gallery>
</LinearLayout>
```

3. Création de notre Activity

Notre activité va donc s'occuper d'initialiser les deux composants. Par défaut, si aucune image n'est trouvée dans la liste, l'image "no_photo" sera affichée. Cette image est placée dans le répertoire drawable de l'application. Le code sera à l'écoute de la sélection d'une image dans la galerie afin d'afficher l'image en grand dans le composant dédié :

Initialisation de l'activité

```
// GUI
private Gallery gallery;
private ImageView imageView;

//Data
private ArrayList<URL> mListImages;
private Drawable mNoImage;

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

    //Récupération d'une image par défaut à
    afficher en cas d'erreur ou de liste vide
    mNoImage =
    this.getResources().getDrawable(R.drawable.no_pho
to);

    //Construction des URL pour les images
    mListImages = buildListImages();

    //Récupération du composant affichant l'image
    en grand
    imageView =
    (ImageView) findViewById(R.id.imageview);

    //On lui met une image par défaut (la première
    de la liste ou, par défaut, l'image d'erreur)
    if (mListImages.size() <= 0) {
        imageView.setImageDrawable(mNoImage);
    } else {
        setImage(imageView, mListImages.get(0));
    }

    //Récupération du composant affichant la liste
    des vignettes
    gallery = (Gallery) findViewById(R.id.gallery);
```

```

//On lui attribue notre Adapter qui s'occupera
de l'alimenter en vignettes
gallery.setAdapter(new AddImgAdp(this));
//Espace entre les vignettes
gallery.setSpacing(10);

//Lors d'un clic sur une des vignettes, on
affiche l'image correspondante en grand
gallery.setOnItemClickListener(new
OnItemClickListener() {
    public void onItemClick(AdapterView parent,
View v, int position, long id) {
        setImage(imgView,
mListImages.get(position));
    }
});
}

```

Vu que la liste des images est assez longue, j'ai créé une méthode pour initialiser cette liste :

Création de la liste des images

```

//Adresse où se trouve l'ensemble des images GIF
(numérotées de 1 à 21).
private final static String SERVER_IM =
"http://mickael-
lt.developpez.com//tutoriels/android/imagesfaq/";

/**
 * Permet de construire la liste des URL pour
les images
 * @return
 */
private ArrayList<URL> buildListImages() {
    int nbTotalImage = 21;
    ArrayList<URL> listFic = new
ArrayList<URL>();
    for(int i = 1; i <= nbTotalImage; i++) {
        try {
            listFic.add(new URL(SERVER_IM + "/" + i +
".gif"));
        } catch (MalformedURLException e) {
            Log.e("DVP Gallery", "Erreur format URL :
" + SERVER_IM + "/" + i + ".gif");
            e.printStackTrace();
        }
    }

    return listFic;
}

```

Ensuite, il nous faut créer un *Adapter* pour gérer l'affichage de notre Gallery. Vous noterez qu'ici les vues sont recyclées afin d'optimiser la mémoire :

Adapter pour l'affichage des vignettes

```

/**
 * Notre Adapter qui gère la liste des
vignettes
 * @author Mickaël Le Trocquer
 */
public class AddImgAdp extends BaseAdapter {
    int GalItemBg;
    private Context cont;

    public AddImgAdp(Context c) {
        cont = c;
        TypedArray typArray =
obtainStyledAttributes(R.styleable.GalleryTheme);

```

```

        GalItemBg =
typArray.getResourceId(R.styleable.GalleryTheme_a
ndroid_galleryItemBackground, 0);
        typArray.recycle();
    }

    public int getCount() {
        return mListImages.size();
    }

    public Object getItem(int position) {
        return position;
    }

    public long getItemId(int position) {
        return position;
    }

    public View getView(final int position, View
convertView, ViewGroup parent) {
        ImageView imgView = null;
        //Récyclage du composant
        if (convertView == null) {
            imgView = new ImageView(cont);
        } else {
            imgView = (ImageView)convertView;
        }
        //On affecte notre image à la vignette
        if (mListImages.size() <= 0) {
            imgView.setImageDrawable(mNoImage);
        } else {
            setImage(imgView,
mListImages.get(position));
        }
        //On définit la taille de l'image
        imgView.setLayoutParams(new
Gallery.LayoutParams(150, 150));
        imgView.setScaleType(ImageView.ScaleType.FI
T_XY);
        //On fixe un arrière-plan plus sympa
        imgView.setBackgroundResource(GalItemBg);

        return imgView;
    }
}

```

Maintenant voici le moyen de télécharger une image à partir d'une URL et de l'affecter à un composant de type *ImageView* :

Téléchargement d'une image et affectation à une vignette

```

/**
 * Méthode permettant de télécharger une image
depuis une URL et de l'affecter à un composant de
type ImageView
 * @param aView
 * @param aURL
 */
public void setImage(ImageView aView, URL aURL)
{
    try {
        URLConnection conn = aURL.openConnection();
        conn.connect();
        InputStream is = conn.getInputStream();

        // Bufferisation pour le téléchargement
        BufferedInputStream bis = new
BufferedInputStream(is, 8192);

        // Création de l'image depuis le flux des
données entrant

```

```

    Bitmap bm =
    BitmapFactory.decodeStream(bis);
    bis.close();
    is.close();

    // Fixe l'image sur le composant ImageView
    aView.setImageBitmap(bm);

} catch (IOException e) {
    aView.setImageDrawable(mNoImage);
    Log.e("DVP Gallery", "Erreur téléchargement
image URL : " + aURL.toString());
    e.printStackTrace();
}
}
}

```

Vu qu'on souhaite télécharger des fichiers depuis Internet, je vous conseille de rajouter l'autorisation dans votre manifest.xml :

```

manifest.xml
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res
/android"
    package="com.dvp.android.gallery"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon"
android:label="@string/app_name">
        <activity android:name=".DVPGallery"
            android:label="@string/app_name"
            >
            <intent-filter>
                <action

```

```

android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" /
                >
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="4" />
    <uses-permission
android:name="android.permission.INTERNET"></uses
-permission>
</manifest>

```

4. Conclusion

Pour conclure, le composant Gallery se gère comme un composant ListView dans son utilisation basique. D'autres paramètres permettent de personnaliser ce composant (espacement entre les items, zoom à appliquer sur l'item ayant le focus, etc). Le téléchargement de ressources sur Internet n'est pas vraiment un problème ici. Le seul point bloquant c'est les temps d'attente pour les téléchargements. Les sources de cette application sont téléchargeables ici : [Lien20](#)
Cet article traite d'un cas simple, pour bien finaliser cet exemple, il faudrait introduire la notion de Thread, d'optimisation pour le téléchargement, etc. Vous pouvez trouver un exemple complet ici : [Lien21](#)

Retrouvez l'article de Mickaël Le Trocquer en ligne : [Lien22](#)

Réaliser un mini navigateur Web

Grâce à ce tutoriel Android, nous allons voir comment utiliser les requêtes HTTP (GET) et les WebView. Les requêtes HTTP de type GET sont intéressantes pour récupérer des informations présentes sur un serveur à partir d'une URL donnée. Les WebViews quant à eux, peuvent faciliter la mise en page de vos applications grâce à des fichiers HTML.

1. Navigateur Web

Nous allons donc réaliser un petit navigateur Web. Grâce à une requête HTTP de type GET, nous allons récupérer le code HTML d'une page et grâce au WebView, nous allons afficher la page correspondante à l'URL que l'on aura saisie.

Créez un projet, nommez-le comme vous le souhaitez, personnellement j'ai utilisé la version 1.6 d'Android.

1.1. Le fichier AndroidManifest.xml

Pour que l'application puisse fonctionner correctement, elle devra avoir accès à Internet. Il faut donc donner l'autorisation d'accès à Internet à l'application. Pour cela, ouvrez le fichier **AndroidManifest.xml** et rajoutez la ligne suivante :

```

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

```

1.2. Le fichier main.xml

Notre interface sera composée d'un **EditText** qui jouera le rôle de la barre d'adresse, d'un **Button** permettant de lancer la requête HTTP en utilisant l'URL que l'on aura indiquée dans l'**EditText** et un **WebView** qui se chargera d'afficher la page Web. Voici donc le fichier **main.xml** :

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        >
        <EditText android:id="@+id/EditText"

```

```

        android:layout_width="wrap_content"
    t"
        android:layout_height="wrap_content"
    nt"
        android:layout_weight="1"
        android:layout_gravity="bottom"
        android:text="http://"
        />

        <Button android:id="@+id/Button"
            android:layout_width="wrap_content"
    t"
            android:layout_height="wrap_content"
    nt"
            android:text="Go"
            />
    </LinearLayout>

    <WebView android:id="@+id/WebView"
        android:layout_height="fill_parent"
        android:layout_width="fill_parent"
        />
</LinearLayout>

```

1.3. Le code JAVA

Comme à mon habitude, j'ai commenté mon code je pense que cela suffira pour comprendre, mais toutes les questions, remarques ou suggestions sont les bienvenues, alors n'hésitez pas à laisser un petit commentaire.

```

package com.tutomobile.android.requetehttp;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URI;

import org.apache.http.HttpResponse;
import
org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import
org.apache.http.impl.client.DefaultHttpClient;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.webkit.WebView;
import android.widget.Button;
import android.widget.EditText;

public class Tutoriel7_Android extends Activity {

    private static final String LOG_TAG = "Log :
";
    private final String mimeType = "text/html";
    private final String encoding = "utf-8";
    private String url;
    private String pageWeb;
    private WebView webView;
    private EditText editText;
    private Button button;

    /** Called when the activity is first

```

```

created. */
    @Override
    public void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //On récupère l'EditText, le WebView, et
le Button grâce au ID
        editText = (EditText)
findViewById(R.id.EditText);
        webView = (WebView)
findViewById(R.id.WebView);
        button = (Button)
findViewById(R.id.Button);

        //On affecte un évènement au bouton
        button.setOnClickListener(
            new OnClickListener() {
                @Override
                public void
onClick(View v) {
                    //créatio
n d'un nouveau Thread pour libérer l'UI Thread le
plus tôt possible (merci tails)
                    new
Thread(){
                        p
                        public void run(){
                            //on récupère l'URL présente dans l'EditText
                            url = editText.getText().toString();

                            try {
                                //on récupère le code HTML associé à l'URL que
l'on a indiquée dans l'EditText
                                pageWeb = Tutoriel7_Android.getPage(url);

                                //on autorise le JavaScript dans la WebView
                                webView.getSettings().setJavaScriptEnabled(true);

                                //on charge les données récupérées dans la
WebView
                                webView.loadDataWithBaseURL("fake://not/needed",
pageWeb, mimeType, encoding, "");
                            } catch (ClientProtocolException e) {
                                e.printStackTrace();
                            } catch (IOException e) {
                                e.printStackTrace();
                            }
                        }
                    }.start()
                }
            });
    }
}

```

```

public static String getPage(String url)
throws ClientProtocolException, IOException{
    StringBuffer stringBuffer = new
StringBuffer("");
    BufferedReader bufferedReader = null;

    try{
        //Création d'un DefaultHttpClient
et un HttpGet permettant d'effectuer une requête
HTTP de type GET
        HttpClient httpClient = new
DefaultHttpClient();
        HttpGet httpGet = new HttpGet();

        //Création de l'URI et on
l'affecte au HttpGet
        URI uri = new URI(url);
        httpGet.setURI(uri);

        //Exécution du client HTTP avec
le HttpGet
        HttpResponse httpResponse =
httpClient.execute(httpGet);

        //On récupère la réponse dans un
InputStream
        InputStream inputStream =
httpResponse.getEntity().getContent();

        //On crée un bufferedReader pour
pouvoir stocker le résultat dans un string
        bufferedReader = new
BufferedReader(new
InputStreamReader(inputStream));

        //On lit ligne à ligne le
bufferedReader pour le stocker dans le
stringBuffer
        String ligneCodeHTML =
bufferedReader.readLine();
        while (ligneCodeHTML != null){
            stringBuffer.append(ligne
CodeHTML);
            stringBuffer.append("\n")
;
            ligneCodeHTML =
bufferedReader.readLine();
        }

    }catch (Exception e){
        Log.e(LOG_TAG, e.getMessage());
    }
}

```

```

}finally{
    //Dans tous les cas on ferme le
bufferedReader s'il n'est pas null
    if (bufferedReader != null){
        try{
            bufferedReader.cl
ose();
        }catch(IOException e){
            Log.e(LOG_TAG,
e.getMessage());
        }
    }

    //On retourne le stringBuffer
return stringBuffer.toString();
}
}

```

1.4. Résultat

Vous pouvez désormais lancer votre application. Entrez une adresse dans la barre d'adresse et cliquez sur "GO". La page Web devrait s'afficher dans le WebView comme sur la capture d'écran ci-dessous.



3. Lien

Tutorial original sur Tuto Mobile : [Lien23](#)

Retrouvez l'article de Axon de Tuto Mobile en ligne : [Lien24](#)

Conférence SUGFR : Concurrent programming and distributed applications with Spring

A l'occasion de sa venue à Paris pour animer une formation, Dave Syer, lead du projet Spring Batch, présentera "Concurrent programming and distributed applications with Spring" dans le cadre du Spring User Group France. Dave a déjà fait cette présentation à SpringOne 2GX, à Chicago, en octobre.

Résumé :

This presentation leads the audience through the minefield of concurrent and distributed computing starting with the basics of Java concurrency, and ending with global patterns for distributed applications. The basic principles of design for such applications are explored and it is shown how using various features of Spring (e.g. task

management, scheduling, POJO adapters) can take the pain out of implementing them in many cases.

Qui est Dave Syer :

Dr David Syer is the technical lead on Spring Batch, the batch processing framework and toolkit from SpringSource. He is an experienced, delivery-focused architect and development manager. He has designed and built successful enterprise software solutions using Spring, and implemented them in major financial institutions worldwide.

La conférence :

La présentation aura lieu le jeudi 9 décembre 2010, à 19h dans les locaux de Zenika, 51, rue Le Peletier à Paris. Vous pouvez vous inscrire à l'adresse suivante : [Lien25](#)

Commentez cette news de Gildas Cuisinier en ligne : [Lien26](#)

Comprendre PDO

De ma perspective personnelle, il m'a semblé que beaucoup de développeurs PHP soient rebutés par PDO. Je dois admettre qu'au tout début, j'étais moi-même dubitatif, et la plupart des tutoriels disponibles m'expliquaient comment faire, mais jamais clairement pourquoi le faire. Ensemble, nous verrons donc quel est le rôle de PDO, en comparaison à `mysql_` et `mysqli_`.

1. Introduction

1.1. Tous ne sont qu'extensions

Saviez-vous que les fonctions ayant le préfixe `mysql_` font partie d'une extension nommée `mysql` ? Eh oui, il en va de même avec `mysqli_...` et PDO aussi d'ailleurs ! À l'heure actuelle, ce sont les trois API disponibles pour établir une connexion à un serveur MySQL avec PHP.

Maintenant que vous avez bien compris qu'il existe trois extensions permettant à PHP de se connecter à MySQL, voici un tableau comparatif :

	Extension MySQL	Extension mysqli	PDO (avec le pilote MySQL Driver et MySQL Native Driver)
Version d'introduction en PHP	Avant 3.0	5.0	5.0
Inclus en PHP 5.x	Oui, mais désactivé	Oui	Oui
Statut de développement MySQL	Maintenance uniquement	Développement actif	Développement actif depuis PHP 5.3
Recommandée pour les nouveaux projets MySQL	Non	Oui	Oui
L'API supporte les jeux de caractères	Non	Oui	Oui
L'API supporte les commandes préparées	Non	Oui	Oui
L'API supporte les commandes préparées côté client	Non	Non	Oui
L'API supporte les procédures stockées	Non	Oui	Oui
L'API supporte les commandes multiples	Non	Oui	La plupart
L'API supporte toutes les fonctionnalités MySQL 4.1 et plus récent	Non	Oui	La plupart

Source : [Lien27](#)

Ce tableau démontre principalement une chose : l'extension MySQL a fait son temps.

1.2. Qu'est-ce que PDO ?

PDO signifie **Php Data Object**. Il s'agit d'une couche d'abstraction des fonctions d'accès aux bases de données. Ça ne signifie donc pas que vos requêtes seront automatiquement compatibles avec n'importe quelle base de données, mais bien que les fonctions d'accès seront universelles :

`mysql_connect()`, `mysql_query()`, `mysql_result()`, `mysql_fetch_array()`, `mysql_real_escape_string()`, etc.

... toutes ces fonctions que vous utilisez sont spécifiques à un SGBD et leur utilisation sera désormais automatiquement déterminée par PDO.

Ainsi, il est temps de casser le mythe laissant sous-entendre que PDO permet une compatibilité entre plusieurs types de bases de données ; PDO n'a pas pour but que d'interpréter vos requêtes et de les traduire pour tous les SGBD. Le but premier de PDO est surtout d'éviter d'apporter une solution au problème de code tel que celui-ci :

Sans PDO

```
switch($typeDb)
{
    case 'mysql':
        mysql_query($query);
        break;
    case 'sqlite':
        sqlite_query($query);
        break;
    case 'mssql':
        mssql_query($query);
        break;
    case 'oci':
        $stid = oci_parse($conn, $query);
        oci_execute($stid);
        //etc...
}
```

En somme, lorsque vous démarrez une connexion PDO, il faudra indiquer quel est le type de SGBD à utiliser. Ensuite, il suffira d'utiliser une fonction unique, fournie par PDO. Pour faire une image, PDO s'occupera de diriger votre requête « vers une fonction d'accès appropriée ». Ça, c'est PDO.

1.3. Pourquoi PDO plutôt qu'un autre ?

Il existe des solutions alternatives à PDO. MDB2, de Pear, en est un bon exemple. La différence majeure est que PDO est une extension compilée de PHP et offre donc des performances supérieures dans la plupart des cas. De plus, des rumeurs fortement insistantes et assez crédibles laissent croire que les futures évolutions de PHP délaisseront progressivement les fonctions standard pour proposer PDO comme solution d'accès aux SGBD par défaut:

Déplacement des extensions de bases de données non PDO vers PECL

[...] Afin de pousser à l'utilisation de cette interface commune, les vieilles extensions PHP seront déplacées vers PECL. Ainsi il ne sera plus possible de faire appel à `mysql_connect()` avec le paquet d'installation par défaut.

Source : [Lien28](#)

D'ailleurs depuis la version 5 de PHP :

PHP 5+ : MySQL n'est plus activé par défaut [...].

Source : [Lien29](#)

1.4. Encore en développement

Vous vous souvenez des cases "La plupart" que PDO avait dans le tableau de l'introduction ? Eh bien oui, c'est que PDO est encore en développement. Bien que PDO ne soit pas nouveau, son utilisation est encore peu répandue. Il est vrai que certains connecteurs (ou pilotes) de PDO sont encore en développement, et que d'autres n'existent pas encore. Voici une liste des connecteurs disponibles ([Lien30](#)), en date du 25 juin 2010 :

- IBM
- Informix
- MySQL
- ODBC et DB2
- PostgreSQL
- SQLite

Les connecteurs suivants sont aussi disponibles, mais expérimentaux :

- 4D
- FireBird/Interbase
- MS SQL (je suis personnellement incapable de transférer des données unicode)
- Oracle

Chaque connecteur peut être ou non activé sur votre serveur sous forme d'extension. Sous Windows, on retrouve donc une DLL pour PDO, en plus d'une DLL pour chaque connecteur. Sous Linux, ce sont des modules `.so`, mais le principe est le même.

Si vous n'êtes pas certain, les fonctions `PDO::getAvailableDrivers()` ([Lien31](#)) et `phpinfo()` ([Lien32](#)), pourront vous indiquer si PDO est installé sur votre serveur, et quels sont les connecteurs disponibles.

2. Les requêtes préparées

2.1. Qu'est-ce qu'une requête préparée ?

Les requêtes préparées ne sont pas propres à PDO, il est possible d'en faire autant avec `mysql_*` (*voir note) qu'avec `mysqli_*`. Le concept est de soumettre un moule de requête et de placer des *place holders* aux endroits où l'on voudra insérer nos valeurs dynamiques. Attention, j'ai bien dit valeurs, et non pas noms de tables, noms de champs, ni même commandes ou constantes internes (DESC, ASC, etc.). Un place holder représente donc une seule et unique valeur.

* En soi, les fonctions `mysql_*` n'offrent aucune fonction supportant les requêtes préparées. Malgré tout, nous verrons qu'il est possible d'en effectuer manuellement depuis toujours (enfin, depuis la version 4.1 de MySQL...).

Voici le principe, vous avez une requête avec des *place holders* (les points d'interrogation) :

```
SELECT *
FROM foo
WHERE id=? AND bar<?
LIMIT ?;
```

Le SGBD va préparer (interpréter, compiler et stocker temporairement son "plan d'exécution logique" (merci doctorrock)) la requête.

Il faudra ensuite associer des valeurs aux *place holders*, qui agissent un peu comme des variables :

```
place holder #1 = 1
place holder #2 = 100
place holder #3 = 3
```

Nous verrons le véritable code pour associer les valeurs plus bas.

Puis, dans une seconde phase, ces valeurs seront soumises en demandant la compilation et l'exécution de la requête qui a été préparée. Le SGBD saura alors que ce qu'elle insère, ce sont des valeurs et non pas des commandes. L'assemblage de la requête, ou sa compilation finale, se fera en interne du SGBD, ce qui explique que vous ne puissiez pas réellement déboguer la valeur compilée de votre requête dans votre code PHP. Dans ce cas-là, la requête exécutée sera donc :

```
SELECT *
FROM foo
WHERE id=1 AND bar<100
LIMIT 3;
```

2.2. PDO != requête préparée

Une des possibilités de PDO, qui est extrêmement populaire, est l'utilisation des requêtes préparées. Beaucoup de personnes semblent confondre cette approche et PDO lui-même. Comme je l'ai affirmé précédemment, PDO a pour but d'offrir une abstraction des fonctions d'accès. En soi, il n'impose pas l'utilisation des requêtes préparées. Comme nous le verrons, PDO n'est pas non plus le seul à permettre les requêtes préparées.

Quoi qu'il en soit, PDO rend l'utilisation des requêtes préparées extrêmement intéressante et accessible, tellement que la plupart des tutoriels passent directement de l'exécution des requêtes avec `mysql_query()`, aux multiples lignes requises pour exécuter une requête préparée dans PDO. Cet article sera certes plus long, mais nous prendrons ensemble le temps de bien faire le pont entre les deux approches...

2.3. Pourquoi les requêtes préparées ?

Soyons bien honnête :

- il vous faudra écrire plus de lignes que pour une requête simple ;
- pour une exécution unique, les performances sont moindres qu'avec une exécution directe ;
- le débogage d'une requête est légèrement plus complexe ;
- et concrètement, le résultat est, à toutes fins pratiques, identique : exécuter une requête.

Alors pourquoi diable a-t-on inventé les requêtes préparées ? Eh bien, exactement pour la même raison que nous avons inventé les systèmes de templates : pour isoler les données du traitement.

Ainsi, l'utilisation de requêtes préparées offrira les avantages suivants :

- impose une certaine rigueur de programmation ;
- optimise le temps d'exécution requis pour les requêtes exécutées plus d'une fois ;
- confère une plus grande sécurité au niveau des requêtes.

2.4. Les requêtes préparées sont plus sécurisées, vraiment ?

Sans prétendre que de ne pas utiliser de requêtes préparées soit un risque, l'argument de sécurité demeure très important. Au sommet du top 10 des risques de sécurité dans les applications Web de l'OWASP pour l'année 2010, se trouvent les failles d'injection. Et la solution recommandée est :

Preventing injection requires keeping untrusted data separate from commands and queries.

Source : [Lien33](#)

Ce qui peut se traduire par : « Prévenir les injections requiert de séparer les données non sûres des commandes et requêtes. »

Et ça tombe bien ; c'est très précisément ce que font les requêtes préparées : séparer les données de la structure de la requête !

3. La pratique, utiliser PDO

3.1. Établir une connexion avec PDO

La première chose à faire pour utiliser PDO est bien sûr d'établir une connexion à une base de données :

Établir une connexion

```
try {
    $strConnection =
    'mysql:host=localhost;dbname=ma_base'; //Ligne 1
    $arrExtraParam=
    array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES
    utf8"); //Ligne 2
    $pdo = new PDO($connStr, 'Utilisateur', 'Mot
    de passe', $arrExtraParam); //Ligne 3; Instancie
    la connexion
    $pdo->setAttribute(PDO::ATTR_ERRMODE,
    PDO::ERRMODE_EXCEPTION); //Ligne 4
}
catch(PDOException $e) {
    $msg = 'ERREUR PDO dans ' . $e->getFile() . '
    L.' . $e->getLine() . ' : ' . $e->getMessage();
    die($msg);
}
```

La première ligne définit la chaîne de connexion (DSN). Il s'agit d'une syntaxe plutôt particulière, débutant par le type de connecteur à utiliser. Ici, `mysql:` indique à PDO d'utiliser son connecteur MySQL. Cette ligne contient aussi les deux autres paramètres : l'adresse du serveur de bases de données, ainsi que la base de données à utiliser.

À la deuxième ligne, nous demandons l'exécution d'une commande pour l'utilisation de l'UTF-8. Nous aurions aussi pu faire une requête standard après l'instanciation de PDO. Vous noterez au passage qu'il est possible de passer les paramètres de connexion via le quatrième paramètre du constructeur, ou par la suite, via la fonction `setAttribute()`.

À noter qu'avec la version 5.3.0 uniquement de PHP, il y a un bogue où la constante `PDO::MYSQL_ATTR_INIT_COMMAND` n'est pas reconnue. Comme alternative, vous pouvez exécuter la requête suivante immédiatement après avoir établi votre connexion :

```
$pdo->query("SET NAMES 'utf8'");
```

Source : [Lien34](#)

La troisième ligne est le véritable début de notre aventure : l'instanciation de la connexion. Si la connexion échoue, PDO lancera une exception.

Finalement, la dernière ligne demande de rapporter les erreurs sous forme d'exceptions. Par défaut, PDO est configuré en mode silencieux; il ne rapportera pas les erreurs. Il existe trois modes d'erreurs :

- **PDO::ERRMODE_SILENT** - ne rapporte pas d'erreur (mais assignera les codes d'erreurs) ;
- **PDO::ERRMODE_WARNING** - émet un warning ;
- **PDO::ERRMODE_EXCEPTION** - lance une exception.

Trouver les chaînes de connexion peut parfois être légèrement difficile. Si les détails sur les DSN dans le manuel de PHP ne vous aident pas, je vous conseille l'excellente contribution de Guillaume Rossolini portant sur les chaînes de connexion (DSN) PDO : [Lien35](#)

Voilà, vous devriez maintenant être en mesure de vous connecter !

Pour la suite du document, nous considérerons que la connexion PDO fut précédemment établie dans la variable

\$pdo, tel que dans l'exemple ci-dessus. De plus, pour des raisons de concision, nous ne placerons pas les try/catch pour chaque exemple, mais en utilisation normale, vous devriez vous assurer de bien gérer vos exceptions, ou alors d'utiliser PDO::ERRMODE_WARNING ou PDO::ERRMODE_SILENT comme mode de gestion des erreurs.

3.2. Sans les requêtes préparées

Si vraiment vous ne voulez pas utiliser les requêtes préparées, vous pouvez utiliser les méthodes query() et exec().

La différence entre ces deux méthodes est que query() retournera un jeu de résultats sous la forme d'un objet PDOStatement, alors que exec() retournera uniquement le nombre de lignes affectées. En d'autres termes, vous utiliserez query() pour des requêtes de sélection (SELECT) et exec() pour des requêtes d'insertion (INSERT), de modification (UPDATE) ou de suppression (DELETE).

Exemple - Effectuer une query et un fetch :

```
PDO
$query = 'SELECT * FROM foo WHERE bar=1;';
$arr = $pdo->query($query)->fetch(); //Sur une
même ligne ...
```

```
mysql_
$query = 'SELECT * FROM foo WHERE bar=1;';
$result = mysql_query($query);
$arr = mysql_fetch_assoc($result);
```

Exemple - Effectuer une query et un fetchAll :

```
PDO
$query = 'SELECT * FROM foo WHERE bar<10;';
$stmt = $pdo->query($query);
$arrAll = $stmt->fetchAll(); //... ou sur 2
lignes
```

```
mysql_
$query = 'SELECT * FROM foo WHERE bar<10;';
$result = mysql_query($query);
$arrAll = array();
while($arr = mysql_fetch_assoc($result))
    $arrAll[] = $arr;
```

Exemple - Effectuer un exec :

```
PDO
$query = 'DELETE FROM foo WHERE bar<10;';
$rowCount = $pdo->exec($query);
```

```
mysql_
$query = 'DELETE FROM foo WHERE bar<10;';
mysql_query($query);
$rowCount = mysql_affected_rows();
```

Petite mise en garde ; un fetchAll() chargera toutes les données en mémoire d'un seul coup. Évidemment, si la quantité de données est particulièrement importante, vous pourriez éprouver des problèmes de performance. Lorsque c'est possible, il est donc préférable de traiter une

seule ligne de résultat à la fois.

Quoi qu'il en soit, dans plusieurs cas, la méthode fetchAll() demeure un raccourci très pratique !

Eh bien, PDO vous fait un peu moins peur maintenant ! Ce n'est finalement pas plus dur que votre façon habituelle de travailler n'est-ce pas ?

3.3. Avec les requêtes préparées

Nous allons maintenant voir comment effectuer des requêtes préparées avec PDO, comparativement à mysqli_ et mysql_.

Pour les requêtes préparées, si le SGBD que vous utilisez ne supporte pas ce type de requêtes, PDO s'occupera d'émuler cette fonctionnalité. La requête sera construite et exécutée comme une requête normale au moment du execute().

Exemple - Effectuer un prepare et un fetchAll:

```
PDO
//Préparer la requête
$query = 'SELECT *
        . ' FROM foo'
        . ' WHERE id=?'
        . ' AND cat=?'
        . ' LIMIT ?;';
$prep = $pdo->prepare($query);

//Associer des valeurs aux place holders
$prep->bindValue(1, 120, PDO::PARAM_INT);
$prep->bindValue(2, 'bar', PDO::PARAM_STR);
$prep->bindValue(3, 10, PDO::PARAM_INT);

//Compiler et exécuter la requête
$prep->execute();

//Récupérer toutes les données retournées
$arrAll = $prep->fetchAll();

//Clôre la requête préparée
$prep->closeCursor();
$prep = NULL;
```

```
mysqli_
//Préparer la requête
$query = 'SELECT *
        . ' FROM foo'
        . ' WHERE id=?'
        . ' AND cat=?'
        . ' LIMIT ?;';
$prep = $mysqli->prepare($query);

//Associer des valeurs aux place holders
$id = 120;
$cat = 'bar';
$limit = 1;
$prep->bind_param('isi', $id, $cat, $limit);
$prep->bind_result($col1, $col2);

//Compiler et exécuter la requête
$prep->execute();

//Récupérer toutes les données retournées
$arrAll = array();
```

```
while($prep->fetch())
    $arrAll[] = array(
        'col1' => $col1,
        'col2' => $col2
    );
```

```
//Clôre la requête préparée
$prep->close();
```

mysql

```
//Préparer la requête
$query = 'PREPARE stmt_name'
        . ' FROM "SELECT *'
        . ' FROM foo'
        . ' WHERE id=?'
        . ' AND cat=?'
        . ' LIMIT ?";';
```

```
mysql_query($query);
```

```
//Associer des valeurs aux place holders
```

```
$query = 'SET @paramId = 120;';
mysql_query($query);
$query = 'SET @paramCat = "bar";';
mysql_query($query);
$query = 'SET @paramLimit = 10;';
mysql_query($query);
```

```
//Compiler et exécuter la requête
```

```
$query = 'EXECUTE stmt_name'
        . ' USING @paramId,'
        . ' @paramCat,'
        . ' @paramLimit;'
```

```
$result = mysql_query($query);
```

```
//Récupérer toutes les données retournées
```

```
$arrAll = array();
while($arr = mysql_fetch_assoc($result))
    $arrAll[] = $arr;
```

```
//Clôre la requête préparée
```

```
$query = 'DEALLOCATE PREPARE stmt_name;';
mysql_query($query);
```

3.4. Réutiliser une requête préparée pour gagner en performance

Outre le fait que vos paramètres sont bien protégés, l'avantage initial des requêtes préparées est la réutilisation du moule de la requête. En effet, le SGBD a déjà effectué une partie du traitement sur la requête. Il est donc possible de réexécuter la requête avec de nouvelles valeurs, sans pour autant devoir reprendre le traitement du départ; le découpage et l'interprétation ont déjà été faits !

Réutiliser une requête préparée

```
$query = 'INSERT INTO foo (nom, prix) VALUES
(?, ?);';
$prep = $pdo->prepare($query);

$prep->bindValue(1, 'item 1', PDO::PARAM_STR);
$prep->bindValue(2, 12.99, PDO::PARAM_FLOAT);
$prep->execute();

$prep->bindValue(1, 'item 2', PDO::PARAM_STR);
$prep->bindValue(2, 7.99, PDO::PARAM_FLOAT);
$prep->execute();

$prep->bindValue(1, 'item 3', PDO::PARAM_STR);
```

```
$prep->bindValue(2, 17.94, PDO::PARAM_FLOAT);
$prep->execute();
```

```
$prep = NULL;
```

Dans ce type de cas de figure qui oblige souvent l'exécution de requêtes dans des boucles, les requêtes préparées représentent une optimisation à ne pas négliger.

4. Les place holders

4.1. Nommage des place holders

Un autre avantage de PDO est qu'il permet l'utilisation de *place holders* nommés, c'est-à-dire d'attribuer un nom au *place holder*, plutôt que d'utiliser des points d'interrogation et un indicateur de position numérique.

Comment utiliser les place holders

```
$query = 'SELECT * FROM foo WHERE id=:id AND
cat=:categorie LIMIT :limit;';
$prep = $pdo->prepare($query);

$prep->bindValue(':limit', 10, PDO::PARAM_INT);
$prep->bindValue(':id', 120, PDO::PARAM_INT);
$prep->bindValue(':categorie', 'bar',
PDO::PARAM_STR);
$prep->execute();

$arrAll = $prep->fetchAll();

$prep->closeCursor();
$prep = NULL;
```

L'ordre d'appel des méthodes `bindValue()` n'a donc plus d'importance.

4.2. Comment faire face à un nombre de place holders dynamique ?

Dans certaines circonstances, le nombre de *place holders* doit être dynamique. Malheureusement, il n'existe pas de façon élégante de procéder. L'exemple suivant illustre bien la problématique et de quelle façon s'y prendre :

Quantité de place holders variable

```
$arr = array(12, 62, 61, 36, 92); //Le nombre
d'éléments est variable.

//Créer une chaîne de place holder
$arrPH = array();
foreach($arr as $elem)
    $arrPH[] = '?';
$strPH = implode(',', $arrPH);
//Contient: ?,?,?,?

//Préparer la requête
$query = 'SELECT * FROM foo WHERE id IN(' .
$strPH . ');';
$prep = $pdo->prepare($query);

//Associer les valeurs aux place holders
for($i=0;$i<count($arr);$i++)
    $prep->bindValue($i+1, $arr[$i],
PDO::PARAM_INT);

$prep->execute();
$arrAll = $prep->fetchAll();
```

```
$prep->closeCursor();
$prep = NULL;
```

... Eh non, la perfection n'est toujours pas de ce monde !

5. Autres cas particuliers

5.1. Trouver le nombre de lignes retournées

Il existe plusieurs cas de figure où il est utile de calculer le nombre de lignes retournées dans un jeu de résultats. Traditionnellement, cela était fait avec la fonction `mysql_num_rows()` pour `mysql_` ou `mysqli_stmt::num_rows()` pour `mysqli`. PDO suit la même logique que `MySQLi`, en ce sens que vous pouvez accéder au compte des lignes via la méthode `PDOStatement->rowCount()` du statement.

Utilisation de `PDOStatement->rowCount()`

```
$query = 'SELECT * FROM foo;';
$prep = $pdo->prepare($query);
$prep->execute();

echo $prep->rowCount() . ' résultat(s)';

//Autre manipulation hors-contexte
$arrAll = $prep->fetchAll();
var_dump($arrAll);
```

Même si vous n'utilisez pas de requête préparée et que vous optez pour la méthode `query()`, celle-ci retourne tout de même un statement PDO, vous avez donc accès à toutes les méthodes de la classe `PDOStatement` sur l'objet retourné par la méthode `query()`.

Utilisation de `PDOStatement->rowCount()`

```
$query = 'SELECT * FROM foo;';
$stmt = $pdo->query();

echo $stmt->rowCount() . ' résultat(s)';

//Autre manipulation hors-contexte
$arrAll = $stmt->fetchAll();
var_dump($arrAll);
```

Il est aussi intéressant de noter que vous ne devriez plus avoir besoin d'utiliser une fonction (ou méthode) de calcul du nombre de lignes pour déterminer si le jeu de résultats est vide ou pas. En effet, si aucun résultat n'est retourné, la méthode `fetch()` retournera `FALSE` et la méthode `fetchAll()` retournera un tableau vide. Je prends la peine de le mentionner car *tous les SGBD n'ont pas nécessairement le même comportement en ce qui concerne le calcul du nombre de lignes retournées*. Autant que possible, il est préférable d'éviter (encore une fois, si possible) d'utiliser ce type d'appel.

Éviter de calculer inutilement le nombre de lignes retournées via `fetch`

```
$query = 'SELECT * FROM foo LIMIT 1;';
$prep = $pdo->prepare($query);
$prep->execute();
$arr = $prep->fetch();

if($arr !== false)
```

```
{
    echo 'Les données sont disponibles: ';
    var_dump($arr);
}
else
{
    echo 'Aucun résultat disponible.';
}
```

Éviter de calculer inutilement le nombre de lignes retournées via `fetchAll`

```
$query = 'SELECT * FROM foo;';
$prep = $pdo->prepare($query);
$prep->execute();
$arrAll = $prep->fetchAll();

if(!empty($arrAll))
{
    echo 'Les données sont disponibles: ';
    foreach($arrAll as $arr)
    {
        echo '<br />Une ligne: ';
        var_dump($arr);
    }
}
else
{
    echo 'Aucun résultat disponible.';
}
```

5.2. Trouver le nombre de lignes affectées

Si vous avez besoin de connaître le nombre de lignes affectées par une requête `INSERT`, `UPDATE` ou `DELETE`, il vous suffit de savoir que ce résultat est transmis comme valeur de retour de la méthode `PDOStatement->rowCount()` dans le cas où vous avez utilisé une requête préparée.

Nombre de lignes affectées par une requête préparée

```
$query = 'DELETE FROM foo WHERE bar=?;';
$prep = $pdo->prepare($query);
$prep->bindValue(1, 6, PDO::PARAM_INT);
$prep->execute();

echo $prep->rowCount() . ' ligne(s) affectée(s).';
```

Si vous détestez toujours autant les requêtes préparées, voici comment trouver ce même résultat via une exécution directe :

Nombre de lignes affectées par une exécution directe

```
$query = 'DELETE FROM foo WHERE bar=6;';
$nbr = $pdo->exec($query);

echo $nbr() . ' ligne(s) affectée(s).';
```

5.3. Explication du faux bogue de la clause `LIMIT`

À leurs débuts avec PDO, beaucoup de gens -- moi inclus -- ont cru qu'il y avait un bogue lorsque l'on utilise un *place holder* dans une clause `LIMIT` ; pourtant, il s'agit bien d'une valeur, alors ça devrait fonctionner !

Par défaut, si le 3e paramètre de la méthode `bindValue()` est ignoré, le type utilisé sera `PDO::PARAM_STR`. Ce qui

signifie que les valeurs auront des guillemets de part et d'autre pour les délimiteurs. En général, ça ne pose pas problème, mais dans le cas de la clause LIMIT, c'est problématique :

Code générant une requête invalide

```
$query = 'DELETE FROM foo LIMIT ?;';  
$prep = $pdo->prepare($query);  
  
$prep->bindValue(1, 10); //Le problème est donc  
à cette ligne  
$prep->execute();
```

Comme aucun type n'est spécifié, PDO gère la valeur 10 comme une chaîne de caractères. La requête réellement exécutée est donc :

Requête invalide générée par le code précédent

```
DELETE FROM foo LIMIT '1';
```

Les guillemets de part et d'autre de la valeur 1 provoquent donc l'erreur suivante :

« *ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near "1" at line 1* »

Voilà, maintenant vous ne pourrez plus dire que PDO c'est moche à cause du non-support des clauses LIMIT. Vous savez maintenant qu'il suffit de définir le 3e paramètre de la méthode bindValue() pour PDO::PARAM_INT !

Code générant une requête valide

```
$query = 'DELETE FROM foo LIMIT ?;';  
$prep = $pdo->prepare($query);  
  
$prep->bindValue(1, 10, PDO::PARAM_INT);  
$prep->execute();
```

6. Conclusion

Si les requêtes préparées ont été introduites avec MySQL 4.1, il faut admettre que leur utilisation était plutôt complexe, longue et surtout pénible. MySQLi a permis de rendre la chose plus accessible, mais somme toute assez complexe.

Si certains préféreront la syntaxe de PDO, l'avantage premier de ce dernier n'est pas pour autant de faire des requêtes préparées en MySQL, mais bien de permettre d'utiliser une syntaxe uniforme dans l'utilisation des fonctions et méthodes d'accès.

N'oubliez pas, lors de l'utilisation des requêtes préparées, vous ne pouvez associer que des valeurs et non des commandes SQL.

Retrouvez l'article de Francois Mazerolle en ligne : [Lien36](#)

Mettez en oeuvre les captchas avec Zend Framework

Cet article va présenter la problématique d'utilisation d'un captcha, puis plusieurs solutions de mise en oeuvre dans un environnement Php/Zend

1. Un captcha, pour quoi faire ?

Dès lors que vous publiez une page Internet, un processus automatisé sera en mesure d'en parcourir toutes les pages, de soumettre tous les formulaires et de suivre tous les liens. Un robot sera ainsi en capacité de se créer un compte sur une application en ligne, de poster du contenu éventuellement malveillant, de spammer les autres membres... bref, de compromettre votre application.

Une rapide réflexion nous amène à la conclusion qu'il faut trouver un moyen de s'assurer qu'un visiteur est bien un visiteur "humain" et non pas un "robot", ce qui revient à établir un test de Turing. La première solution qui a permis, dans une certaine mesure, de valider ce test était de poser une question simplissime au visiteur, comme le résultat d'une opération mathématique $1+3=?$. En générant une opération mathématique simple et aléatoire pour chaque visiteur, en la stockant en session avec son résultat, on était en mesure de vérifier que la soumission d'un formulaire possédait la bonne réponse et donc qu'elle avait été initiée par un être humain.

Ce premier essai montre cependant très vite ses limites puisqu'un robot est capable de repérer cette protection dans un formulaire, d'interpréter cette opération mathématique, de la résoudre et de renvoyer le résultat dans un formulaire, se faisant passer pour plus humain qu'il n'est. De la même manière on s'est rendu compte que toute question basée sur une information textuelle pouvait être récupérée et résolue automatiquement.

L'idée maîtresse est de fournir un "objet" ne pouvant être interprété que par un humain. Et pour faciliter la vérification de cet "objet" par votre application, l'idée est de représenter du texte de manière suffisamment déformée. Dans cette description assez vague, on retrouve le texte mis en image et le texte mis en ASCII art, deux propositions que je vais présenter.

Certes vous pourriez essayer de générer vous-mêmes des captchas en manipulant une image, en rendant un texte aléatoire dans cette image, en ajoutant des déformations et du bruit, mais ces opérations sont déjà existantes dans Zend.

2. Zend_Captcha

Le framework Zend fournit plusieurs solutions de gestion et de vérification de Captcha. Zend amène les objets de base pour gérer les captchas, mais amène aussi de quoi les intégrer dans un formulaire Zend_Form. Tous les types de captchas gérés par Zend implémentent

Zend_Captcha_Adapter et peuvent facilement être intégrés dans un Zend_Form.

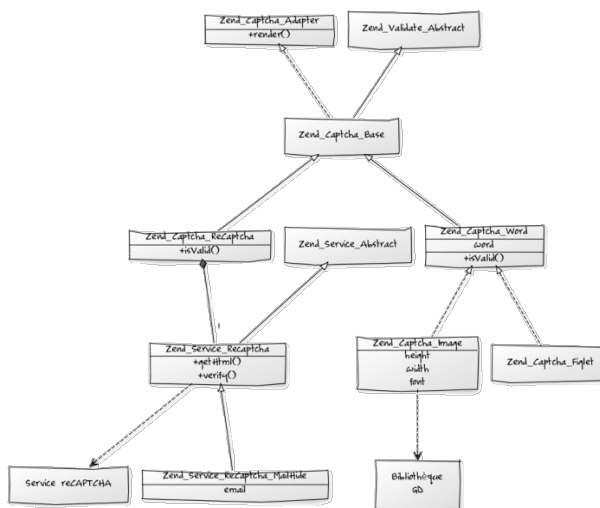
Zend Framework propose des solutions pour gérer des captchas de plusieurs types :

- un mot à repérer dans une image déformée ;
- un mot représenté en ASCII art ;
- un mot à renvoyer inversé ;
- un mot à repérer dans un captcha recaptcha.

La récupération du mot inversé n'est pas une technique fiable pour valider un test de Turing, nous ne nous y étendrons pas.

Je vais présenter des cas d'utilisation pour les trois autres types de captchas gérés par Zend. Implémentant les mêmes interfaces, leur utilisation sera très proche. Comme pour tous les types Zend_Captcha_*, le fonctionnement reste le même : instanciation du captcha avec le bon paramétrage, insertion du captcha dans le code de rendu html, vérification lors de la soumission d'une requête.

Voilà le schéma UML d'organisation des différents types de captchas Zend :



Voilà le schéma global des exemples que je vais présenter

```
<html>
<head></head>
<body>

<?php
require_once 'Zend/Loader/Autoloader.php';
```

```

Zend_Loader_Autoloader::getInstance()->
registerNamespace('Zend_');

// instantiation du captcha à utiliser
$captcha = new ...
// paramétrage du captcha
$captcha-> ...

if (!isset($_POST['captcha'])) {
    // premier affichage, on génère un
    captcha

    $captcha->generate();
} else {

    // une réponse est proposée pour la
    résolution du captcha

    // vérification du captcha
    if ($captcha->isValid( ... )) {
        echo '';
        $captcha->generate();
    } else {
        echo '';
        $captcha->generate();
    }
}
?>

<form method="post">

<?php echo $captcha->render () ?>

<input type="hidden" name="captcha[id]" value="<?
php echo $captcha->getId() ?>" size="40" /><br/>
<input type="text" name="captcha[input]"
size="40" /><br/>
<input type="submit"/>

</form>

</body></html>

```

2.1. Zend_Captcha_Figlet

Cet objet, bien qu'anecdotique, permet de créer des captchas où le mot à repérer se trouve écrit en ASCII art. Voilà des exemples de rendus que l'on pourra obtenir :



Le développeur ne pourra pas paramétrer grand-chose : à part le nombre de lettres du captcha, rien de spectaculaire.

```

// instantiation et paramétrage
$captcha = new Zend_Captcha_Figlet();
$captcha ->setWordLen(7);

```

2.2. Zend_Captcha_Image

Voilà vraiment la classe qui permettra de générer des images déformées pour masquer du texte. Cet objet intègre du texte dans une image et y ajoute du bruit : des étoiles et des lignes. Le développeur a la possibilité de spécifier la taille (hauteur, largeur) de l'image générée, la police de caractères à utiliser, la taille du texte, le niveau de bruit des étoiles et le niveau de bruit des lignes.

Zend_Captcha_Image utilise la bibliothèque GD pour générer les images, vous devez donc l'avoir installée sur le serveur web. Etant donné que les captchas seront générés puis stockés directement sur le serveur, cette solution de captchas est parfaitement utilisable dans le cas où le client ne serait présent que sur un intranet sans accès extérieur. En ce sens c'est la solution la mieux maîtrisée puisque le serveur a toutes les clefs : la génération du captcha et sa validation.

Voici quelques exemples de captchas faisant varier le niveau de bruit des étoiles et des lignes :

	lignes : 0	lignes : 2	lignes : 20
points : 0	wym8v	qab7c	iu0a8
points : 20	j6u28	voxuz	g9u6h
points : 100	viru6	bys2w	ub32n
points : 1000	32593	7013h	4301

Attention cependant au choix de la police. Bien que le but d'un captcha soit de masquer un texte et de le rendre peu (auto)lisible, il ne faut pas tomber dans l'excès d'un camouflage trop important. Il suffit de générer un captcha avec une police cursive (Mistral par exemple) pour obtenir des captchas vraiment peu lisibles même par un être humain. Bon courage :



Les autres paramètres du Zend_Captcha_Image concernent l'emplacement de stockage des captchas générés et n'offrent que peu d'intérêt pédagogique. Voici le code illustrant l'utilisation de Zend_Captcha_Image, toujours selon notre template défini plus haut.

```

<?php

require_once 'Zend/Loader/Autoloader.php';
Zend_Loader_Autoloader::getInstance()-
>registerNamespace('Zend_');

// instantiation, paramétrage
$captcha = new Zend_Captcha_Image();
$captcha ->setWordLen(8)
            ->setHeight(100)
            ->setWidth(300)
            ->setFont("./tahoma.ttf")
            ->setFontSize(50)
            ->setSuffix(".png")
            ->setImgDir("out/")
            ->setImgUrl("out")
            ;

if (!isset($_POST['captcha'])) {
    $captcha->generate();
} else {

```

```

// vérification

if ($captcha->isValid($_POST['captcha']))
{
    echo '';
    $captcha->generate();
}
else{
    $captcha->generate();
    echo '';
}
}
?>
<form method="post">

<br/>

<input type="hidden" name="captcha[id]" value="<?
php echo $captcha->getId() ?>" size="40" /><br/>
<input type="text" name="captcha[input]"
size="40" /><br/>
<input type="submit"/>
</form>

```

Toutes les images générées seront ici stockées dans le dossier out/. Zend_Captcha_Image intègre une logique de garbage collector et passe automatiquement régulièrement pour vider le contenu de ce dossier. La fonction setGcFreq permet de spécifier à quelle fréquence de requêtes le garbage collector doit être appelé.

On notera que la vérification du captcha nécessite l'id du captcha généré ainsi que le mot repéré qui doivent être passés dans un même paramètre HTTP.

2.3. Zend_Captcha_reCaptcha

Ce système de captcha utilise le service web reCAPTCHA (racheté par Google) pour générer et vérifier le texte à reconnaître. Les images proposées à la reconnaissance de mots proviennent d'éléments non reconnus dans une analyse automatique de caractères. En résolvant des captchas reCAPTCHA, vous participez à la numérisation d'ouvrages qui n'ont pu être entièrement numérisés. Vous avez toujours deux mots à reconnaître : l'un d'eux est connu (c'est sur celui-ci que se fera la validation) et l'autre est à déchiffrer : c'est la participation à la numérisation.

Pour pouvoir utiliser le service reCAPTCHA, vous devez créer des clés d'authentification publique et privée. Pour cela rendez-vous sur le site <https://www.google.com/recaptcha/admin/create> ([Lien37](#)) puis générer une paire de clés pour le domaine Web de votre application. Il n'est pas rare de devoir générer une paire de clés en mode développement si vous travaillez sur un domaine *.localhost et une autre paire de clés pour la mise en production effective de votre application.

L'utilisation de reCAPTCHA nécessite que le client ait accès à Internet puisque les captchas seront directement fournis par le service reCAPTCHA. Le serveur devra lui aussi être relié à Internet puisque c'est lui qui demandera à reCAPTCHA de vérifier si un mot donné résout ou non le captcha. Dans cette solution, le serveur ne fait ni la génération du captcha ni sa vérification, il ne connaît jamais les mots cachés dans le captcha.

Comme pour les méthodes précédentes, la mise en oeuvre est très simple :

```

<html>
<head>
</head>
<body><?php

require_once 'Zend/Loader/Autoloader.php';
Zend_Loader_Autoloader::getInstance()-
>registerNamespace('Zend_');

// instantiation
$captcha = new Zend_Service_ReCaptcha(
    "votre clef publique",
    "votre clef privée"
);
// paramétrage
$captcha->setOptions(array(
    'theme'=>'clean',
    'lang'=>'fr'
));

if (!isset($_POST['recaptcha_challenge_field'])) {
}
else{
    // vérification
    $captchaResult = $captcha->verify(
        $_POST['recaptcha_challenge_field'],
        $_POST['recaptcha_response_field']
    );

    if ($captchaResult->isValid()){
        echo '';
    }
    else{
        echo '';
    }
}
?>
<form method="post">

<?php echo $captcha->getHtml () ?>

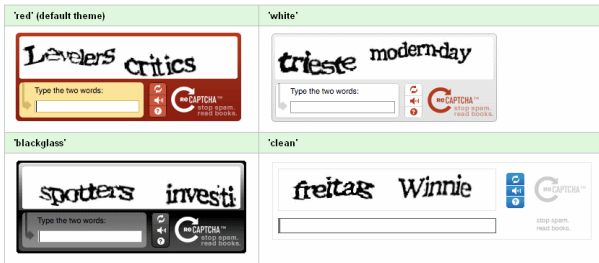
<input type="submit"/>

</form>
</body>
</html>

```

On remarque que la vérification se fait sur des champs POST nommés recaptcha_challenge_field et recaptcha_response_field. Ces deux champs sont positionnés par l'outil reCAPTCHA et sont générés lors de l'appel à \$captcha->getHtml(). Ce n'est pas à vous de les générer.

Le développeur n'a pas beaucoup de possibilités pour paramétrer le captcha reCAPTCHA : à part le look dudit captcha, pas grand-chose. Voici les quatre looks existants :



2.4. Captchas avec Zend_Form

Tous les captchas gérés par Zend implémentent `Zend_Captcha_Adapter`, ils peuvent ainsi tous être ajoutés à un `Zend_Form` de la même manière via un `Zend_Form_Element_Captcha`. Exemple :

```
$form = new Zend_Form();

// construction du Zend_Form avec tous les champs
nécessaires
$form->addElement(...);
...

// création du captcha
$captcha = new
Zend_Form_Element_Captcha('captcha', array(
    'label' => "Merci de confirmer que vous êtes
humain",

    // paramétrage en reprenant les noms de
```

```
méthodes vus précédemment
'captcha' => array(
    "captcha" => "Image",
    "wordLen" => 8,
    "font" => "./tahoma.ttf",
    "height" => 100,
    "width" => 300,
    "fontSize" => 50,
    "imgDir" => "out/",
    "imgUrl" => "out/"
)
));

$form->addElement($captcha);

// suite du rendu du Zend_Form
```

La validation du formulaire se fera via `$form->isValid($_POST)`, l'utilisation du captcha devient complètement transparente, le développeur n'a plus à la gérer manuellement.

3. Conclusion

Zend fournit les outils nécessaires à une gestion de captchas qui devrait convenir aux utilisations les plus courantes. Mais bien sûr, rien ne vous empêche de développer votre propre captcha héritant de `Zend_Captcha_Base` si vous ne trouvez pas votre bonheur.

Retrouvez l'article de Pierre Schwartz en ligne : [Lien38](#)

L'API géolocalisation en HTML5

Une des nouveautés introduites par HTML5 est la géolocalisation utilisable via une API d'un navigateur. Cela permet aux pages Web d'interroger le navigateur sur la position géographique de l'utilisateur.

L'API de base permet d'obtenir les coordonnées en latitude et en longitude ainsi que l'altitude.

Celles-ci peuvent alors être exploitées par le biais d'une carte (de type Google Map).

1. Compatibilité

Chrome, Safari, Opera, Firefox.

2. Utilisation de l'API

Comment obtenir la position courante de l'utilisateur ?
Voici la marche à suivre :

Tout d'abord, on vérifie que l'API est disponible ;

```
if(navigator.geolocation){  
  ...  
}
```

Ensuite, on cherche la position (attention le navigateur vous demandera la permission, il faut l'accepter).

Il faut noter que l'altitude n'est disponible que sous Firefox.

```
if(navigator.geolocation){  
  navigator.geolocation.getCurrentPosition(function(position){  
    var latitude = position.coords.latitude;  
    var longitude = position.coords.longitude;  
    var altitude = position.coords.altitude;  
    document.getElementById('geolocation').innerHTMLHTML = 'latitude : ' + latitude + '<br />' +  
    'longitude : ' + longitude + '<br />' + 'altitude : ' + altitude + '<br />';  
  });  
}
```

Démo : [Lien39](#)

La fonction **navigator.geolocation.getCurrentPosition()** prend en argument une fonction dans laquelle on peut utiliser la variable position qui renvoie les coordonnées.

navigator.geolocation.getCurrentPosition() peut également prendre en argument une fonction de gestion d'erreur et des options (dans un objet).

Exemple :

```
navigator.geolocation.getCurrentPosition(  
  successfunction, errorfunction, {options}  
);
```

3. Les options

Les options peuvent être les suivantes (si elles sont implémentées dans le navigateur utilisé) :

- **enableHighAccuracy** : (true ou false), position précise ou non ;
- **timeout** : (type long), temps de réponse, durée avant renvoi vers la fonction d'erreur ;
- **maximumAge** : (type long ou Infinity) durée de la mise en cache de la position courante, si *maximumAge:0* alors la position ne viendra jamais du cache, elle sera toujours renouvelée.

```
function errorfunction(error){  
  switch(error.code) {  
    case error.TIMEOUT:  
      //faire quelque chose pour la gestion du timeout  
      break;  
  }  
}  
function successfunction(position){  
  if (position.timestamp < freshness_threshold &&  
  position.coords.accuracy < accuracy_threshold) {  
    // la position est relativement fraîche et précise.  
  } else {  
    // la position est ancienne ou imprécise.  
  }  
}  
navigator.geolocation.getCurrentPosition(  
  successfunction, errorfunction,  
  {maximumAge:5000, timeout:2000}  
);
```

4. L'objet position

position retourne les coordonnées mais également d'autres valeurs :

- **coords**, retourne les coordonnées ainsi que la précision (entre autres) ;
- **timestamp**, représente le moment où la position a été acquise.

coords retourne plusieurs valeurs :

- **latitude**, la latitude de la position ;
- **longitude**, la longitude de la position ;
- **altitude**, l'altitude de la position ;
- **accuracy**, niveau de précision de la longitude et de la latitude (en mètre) ;
- **altitudeAccuracy**: niveau de précision de

- l'altitude (en mètre) ;
- **heading**, donne la position en degré par rapport au nord ;
- **latitude**, affiche la vitesse actuelle de déplacement de la position (en mètre).

Bien entendu, l'entièreté de ces valeurs n'est pas encore présente dans tous les navigateurs.

Les plus répandues sont : **latitude**, **longitude** et **altitude** (unique à Firefox pour l'instant). Si une valeur n'est pas existante, elle retournera **null**.

5. Fonction d'error : PositionError

Lorsque la fonction de callback d'erreur est appelée, elle a pour argument un objet de type **PositionError**

Voici la liste des propriétés de l'objet :

- **UNKNOWN_ERROR**, erreur inconnue ;
- **PERMISSION_DENIED**, l'application n'a pas l'autorisation d'utiliser l'API Geolocalisation ;
- **POSITION_UNAVAILABLE**, la position n'existe pas ;
- **TIMEOUT**, erreur de timeout ;
- **code**, renvoie le code d'erreur courant ;
- **message**, renvoie le message.

```
function errorfunction(error) {
    switch(error.code) {
        case error.TIMEOUT:
            //faire quelque chose pour la gestion du
            timeout
            break;
        case error.PERMISSION_DENIED:
            alert(error.message);
            break;
    }
}
function successfunction(position) {
    ...
}
```

```
navigator.geolocation.getCurrentPosition(
    successfunction, errorfunction,
    {maximumAge:5000, timeout:2000}
);
```

6. Les fonctions watchPosition() et clearWatch()

watchPosition() est une fonction asynchrone qui retourne une position immédiatement et entame un processus de veille. Si la position de l'utilisateur change (sur un mobile par exemple), cette fonction retournera immédiatement une valeur de position.

Cela ressemble un peu à un `setInterval` (très grossièrement), mais sur une position.

Enfin pour supprimer ce processus de veille entamé par **watchPosition()**, on utilisera **clearWatch()** qui prend en argument l'adresse de **watchPosition()**.

Voici un exemple pour illustrer ce propos :

```
var watchId =
navigator.geolocation.watchPosition(
    successfunction, errorfunction,
    {maximumAge:5000, timeout:2000}
);

function cancelButton(){
    clearWatch(watchId);
}
```

7. Exemple concret avec Google Map

Démo : [Lien40](#)

8. Conclusion

En conclusion, on remarquera que la localisation n'est pas toujours très fiable car elle se base sur l'adresse IP des machines (ou des proxys !). On peut donc penser que cette API sera plus utilisée pour les applications Web orientées mobile.

Retrouvez l'article de Jérôme Debray en ligne : [Lien41](#)

Comprendre le storage en HTML5

La fonctionnalité storage est une nouveauté HTML5, ce procédé va permettre de simplifier le processus de sauvegarde et de persistance des données.

Cette fonctionnalité est similaire aux cookies de session HTTP et permet de sauvegarder les données voulues dans une "base de données" côté client (c'est-à-dire au niveau du navigateur).

Si vous voulez voir le contenu de `sessionStorage` et `localStorage`, avec Chrome, c'est très facile via les "outils de développement", onglet "Storage".

1. Compatibilité

Chrome, Safari, Opera, Firefox 4, Internet Explorer 9

2. Intérêt du storage

- On peut enregistrer des données même quand la connexion Internet est coupée (le **storage** étant dans le navigateur).
- On peut gérer des données par session (si vous avez deux fenêtres sur le même site, une action

sur le **sessionStorage** sera répercutée sur l'autre).

- On peut gérer des données par page (**localStorage**).

Le **storage** a pour but de répondre à une des limites des cookies, c'est-à-dire leur taille (4 ko pour les cookies et jusqu'à **10Mo** pour le storage !).

Nous allons donc voir à travers des exemples le **localStorage** et le **sessionStorage** via l'API **storage** en **HTML5**.

3. L'API storage

Chaque objet **storage** permet d'accéder à une liste de clés/valeurs (items). Les clés sont des string (la clé peut être vide).

Attribut

length : retourne le nombre de paires clé/valeur.

Méthode

key(n) : retourne la clé à l'index n, retourne NULL s'il n'y a pas de résultat.

getItem(key) : retourne la valeur de la clé key.

Il est à noter que le storage est un tableau, on peut accéder aux données de cette manière :

Exemple :

```
alert(window.sessionStorage['cle']);
```

setItem(key, value) : ajoute une paire clé/valeur.

Il est à noter que le storage est un tableau, on peut donc attribuer une paire clé/valeur de cette façon :

Exemple :

```
window.sessionStorage['cle'] = 'ma valeur';
```

Si la clé existe déjà la valeur sera mise à jour.

Le **setItem** peut retourner une exception :

- **NOT_SUPPORTED_ERR** : la valeur n'est pas supportée;
- **QUOTA_EXCEEDED_ERR** : il n'y a plus de place pour mémoriser d'autres valeurs ou alors l'utilisateur a **désactivé** le **storage**.

En cas d'échec et donc d'exception la paire clé/valeur ne sera pas créée ni mise à jour (si elle était existante).

removeItem(key) : supprime la paire clé/valeur dont la clé est **key**. Si la clé n'existe pas, rien n'est effacé.

clear() : vide toutes les paires clé/valeur, s'il y en a.

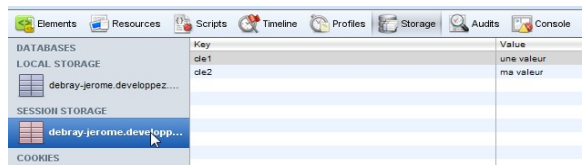
4. sessionStorage

Commençons par un exemple :

```
if(window.sessionStorage){
    window.sessionStorage.setItem('cle1', 'une
valeur');
    /*Autre manière de le faire*/
    window.sessionStorage['cle2'] = 'ma valeur';
}else{
```

```
$('.testSession').html('le sessionStorage
n'est pas implémenté sur ce navigateur');
}
```

On ajoute deux paires de clé/valeur dans le **sessionStorage**. Voici le résultat sous Chrome :



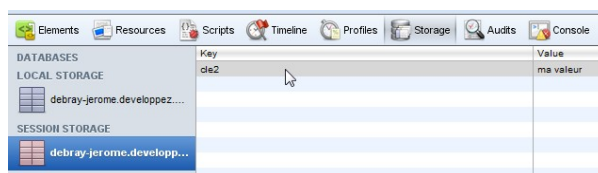
Key	Value
cle1	une valeur
cle2	ma valeur

Démo : [Lien42](#)

Maintenant, voyons le **removeItem**

Tout d'abord, affichez toutes les données et ensuite effacez-en une, puis réaffichez toutes les données : [Lien43](#)

Le résultat du **removeItem** est visible dans l'onglet **Storage** des outils développeurs de Chrome :



Key	Value
cle2	ma valeur

5. localStorage

le **localStorage** fonctionne de la même manière que le **sessionStorage** :

```
window.localStorage.getItem('key');
window.localStorage.setItem('key', 'value');
window.localStorage.removeItem('key');
window.localStorage.length;
window.localStorage.key(n);
```

Exemple :

```
if(window.localStorage){
    window.localStorage.setItem('cle1',
Math.random());
}
```

6. La différence entre localStorage et sessionStorage

La différence entre les deux se trouve dans la durée de stockage des données. **sessionStorage** persiste le temps d'une session de navigation. Ouvrir une nouvelle fenêtre ou un nouvel onglet, initialisera donc un nouvel objet. On pourrait donc utiliser **sessionStorage** pour un formulaire multi page, et conserver de l'information d'une page à l'autre. **localStorage**, quant à lui, conserve les données pour une durée indéterminée. On accède uniquement aux paires clé/valeur d'un même domaine.

Retrouvez l'article de Jérôme Debray en ligne : [Lien44](#)

Zoomer une image façon thumbnail en CSS

Zoomer une image avec le langage CSS... fastoche. Faire des vignettes zoomables façon "thumbnail" demande un peu plus de dextérité. Ce tutoriel expliquant comment créer une série de vignettes zoomables à la sauce 100% CSS recommande d'avoir un niveau débutant averti.

1. Première technique

Fonctionne avec :

- FireFox ;
- Mozilla / SeaMonkey ;
- MSIE 6+ ;
- Opéra 7+ ;
- Safari.

Attributs utilisés :

- background-color ;
- color ;
- display ;
- height ;
- width ;
- margin ;
- padding ;
- position ;
- top ;
- left.

1.1. Code (X)HTML

Les images sont déclarées dans le code HTML sans valeur de taille (ni width, ni height), chacune d'elles est dans un `<div class="thumb">` dont les propriétés seront déclarées dans la feuille de style. Une seule image par zoom est nécessaire, le poids de la page doit pouvoir rester raisonnable si l'on n'utilise pas d'image trop grande.

```
<div class="thumb">
  <a href="#">
    
  </a>
</div>
<div class="thumb">
  <a href="#">
    
  </a>
</div>
<div class="thumb">
  <a href="#">
    
  </a>
</div>
```

1.2. Code CSS

J'ai choisi de présenter les photos sur un fond noir déclaré dans body. La hauteur de 500px n'est là que pour faire apparaître un ascenseur vertical en cas de résolution d'écran inférieure à 1024*768, et dont l'absence peut rendre difficile la visualisation des photos agrandies. Puis on fixe une bonne fois pour toutes la largeur des bords des images à zéro.

Les vignettes déclarées dans la classe .thumb ont une taille de 100*75 pixels. Je désire qu'elles flottent les unes par rapport aux autres à gauche (float:left), qu'elles soient un petit peu espacées (margin:1px). Lorsqu'elles disparaîtront au profit de l'image zoomée, il apparaîtra à la place un rectangle gris (background-color:#D3D3D3;). Et surtout, pour éviter des sauts intempestifs, on les fixe grâce au display:block.

Vient ensuite le comportement que doivent avoir ces vignettes lors du survol par le curseur de la souris, et c'est là que ça se corse.

Tout d'abord il va falloir s'affranchir des différences de comportements entre MSIE et Mozilla. On commence par stabiliser le tout avec (encore...) un display:block pour les liens.

Il va falloir ensuite avoir recours à un "hack". Pour MSIE, il faut que les liens soient déclarés en position absolue, mais pour Mozilla, il faut que ce soit en position relative sinon le rollover ne fonctionne pas de gauche à droite... Ceci explique le body > .thumb a:hover que MSIE n'interprète pas mais qui donnera à Mozilla la bonne définition.

Il est ensuite nécessaire de déclarer la taille des images liées : .thumb a img donne donc au navigateur les instructions nécessaires à la taille des images mises en lien dans le `<div class="thumb">`.

Et enfin, on déclare la taille et la position que devront avoir les images zoomées. La position est donc relative aux vignettes, et la taille de 300*225px.

Vous avez tout suivi ? Tout compris ? Non ?!! Ah flûte... Voilà le code quand même :

```
body {
  background-color:black;
  color:white;
  height:500px;
}

img {
  border:0
}

.thumb {
  width:100px;
  height:75px;
  margin:1px;
  float:left;
  background-color:#D3D3D3;
  display:block;
}
```



```

.thumb a {
  display:block;
}

.thumb a:hover {
  position:absolute;
}

/*hack pour permettre le rollover
de gauche à droite avec mozilla*/

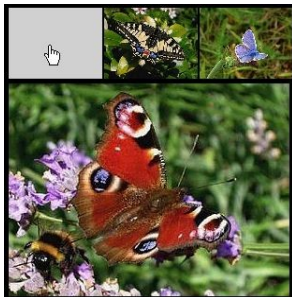
body > .thumb a:hover {
  position:relative;
}

.thumb a img {
  margin:0;
  padding:0;
  width:100px;
  height:75px;
}

.thumb a:hover img {
  position:relative;
  left:0px;
  top:80px;
  width:300px;
  height:225px;
}

```

Résultat :



Voir le résultat en ligne : [Lien45](#)

2. Seconde technique

Fonctionne avec :

- FireFox ;
- Mozilla / SeaMonkey ;
- MSIE 6+ ;
- Opéra 7+ ;
- Safari.

Attributs utilisés :

- background-color ;
- color ;
- display ;
- height ;
- width ;
- margin ;
- position ;
- top ;
- left ;
- text-decoration;

La première technique fait disparaître la vignette au profit de l'image zoomée et celle-ci se décale au fur et à mesure vers la droite. La vignette ne disparaît pas avec cette seconde technique, et les images zoomées apparaissent toujours au même endroit. Par contre, le poids de la page sera plus élevé car dans le code (X)HTML on déclarera la vignette et l'image zoomée.

Pour cet exemple, il y aura donc six images en tout contre trois précédemment.

2.1. Code (X)HTML

```

<div class="thumb">
  <a href="#">
    
    
  </a>

  <a href="#">
    
    
  </a>

  <a href="#">
    
    
  </a>
</div>

```

Comme vous le constatez, il n'y a cette fois qu'un seul cadre (<div class="thumb">), par contre une autre classe est nécessaire pour les photos zoomées (class="grand").

2.2. Code CSS

Même topo que précédemment pour les attributs de body, par contre un bord de 1 pixel noir est déclaré pour les images, mais on peut s'en passer, c'est juste une histoire de goût.

La classe .thumb définit une position relative du cadre en haut à gauche (à partir de ce qu'il y a juste au-dessus). Dans .thumb a on fixe les marges à zéro et le non-soulignement des liens pour éviter tout parasite. Et alors là, attention : pour que le survol puisse prendre corps, il faut absolument déclarer quelque chose pour les liens survolés, quelque chose de neutre... un fond de couleur noire par exemple (thumb a:hover).

On passe maintenant aux caractéristiques des images zoomées.

La classe .grand est appliquée aux images liées, incluses elles-mêmes dans la classe thumb (ah ben oui, je sais, ça fait un peu poupées gigognes tout ça...). Donc, dans .thumb a .grand on aura un display:block et une position absolue (sinon, tout ce petit monde va aller se balader n'importe où), et surtout, une largeur de 0px pour éviter des chevauchements intempestifs.

Il n'y a plus qu'à s'occuper de la taille et de la position de l'image zoomée au moment du survol du curseur : position

absolue, à 80px en dessous de la position initiale (c'est-à-dire du haut des vignettes), complètement à gauche, et de taille 300*225 pixels.

OUF ! Mal à la tête ? C'est normal, c'est le métier qui rentre. Bon, en même temps ce n'est pas très grave si vous n'avez pas tout compris, moi non plus de toute façon...

```
body {
  background-color:black;
  color:white;
  height:500px;
}

img {
  border:1px solid black;
}

.thumb {
  position:relative;
  top:0;
  left:0;
}

.thumb a {
  margin:0;
  text-decoration:none;
}

.thumb a:hover {
  background-color:black;
}
```

```
.thumb a .grand {
  display:block;
  position:absolute;
  width:0;
}

.thumb a:hover .grand {
  position:absolute;
  top:80px;
  left:0;
  width:300px;
  height:225px;
}
```

Résultat :



Voir le résultat en ligne : [Lien46](#)

C'était un peu plus dur, mais ça valait le coup non ?

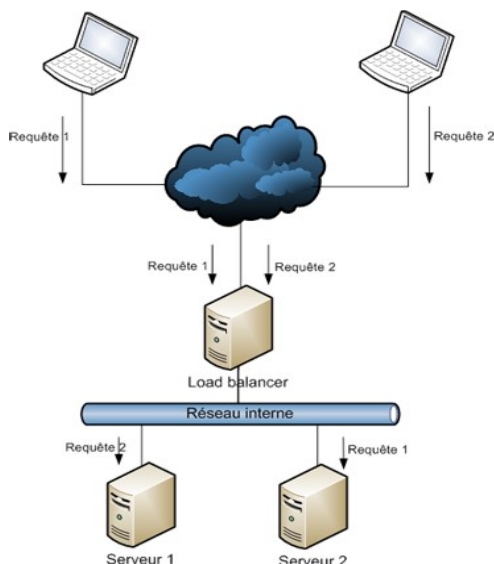
Retrouvez l'article de Pascale Lambert en ligne : [Lien47](#)

Load balancing avec JBoss et Apache 2

Les applications réparties étant de plus en plus diffusées auprès des partenaires, des clients et des collaborateurs, elles se doivent de tenir des charges toujours plus importantes. Le seul matériel ne peut répondre à un coût raisonnable. C'est là qu'intervient le "load balancing" en répartissant la charge entre plusieurs serveurs plus modestes. Ce tutorial doit vous amener à entrevoir ce que ces solutions peuvent vous apporter dans un cas très concret.

1. Introduction

Suite à la formation JBoss plusieurs voies d'approfondissement s'offrent à moi mais il y en a une qui me tient plus à coeur : le **load balancing**. Pourquoi cela? Après tout en tant qu'Architecte logiciel, certains considèrent que notre problématique est de savoir que cela existe. La technique restant le domaine de prédilection des administrateurs et architectes SI. Personnellement ce n'est pas ma conception. Même si je ne prétends pas avoir les mêmes compétences, j'aime savoir de quoi je parle.



Tout d'abord une petite définition du load balancing : le « **load balancing** » ou « **répartition de charge** » est une technique utilisée en informatique pour distribuer un travail entre plusieurs processus, ordinateurs, disques ou autres ressources (source Wikipédia).

Le load balancing peut se faire de différentes manières :

- **matériel** : solution permettant de résister à la plus forte charge. Elle aura des problèmes en cas de répartition sur un protocole haut niveau utilisant les sessions. Cette solution a un coût élevé ;
- **logiciel** : cette solution permet de gérer facilement des répartitions de charge avec respect de session et a un coût faible. Plus lent que le hard, elle dépend du serveur qui l'héberge pour définir ses performances.

Pour ce tutorial, nous allons nous intéresser uniquement à

la partie logicielle qui est plus simple à mettre en oeuvre. Nous utiliserons pour ce faire la configuration la plus courante dans le monde Java :

- deux serveurs d'applications JBoss (avec des configurations différentes mais une application commune habituellement les deux serveurs seront totalement iso) ;
- un serveur Apache 2 sur lequel nous activerons le module mod_jk.

2. Préparation

Je vous conseille de commencer par créer un répertoire où vous installerez les différents serveurs nécessaires.

2.1. Installation JBoss

Pour ce tutorial, j'ai utilisé une version 6.0M4 de JBoss que vous pouvez télécharger sur le site officiel de JBoss ([Lien48](#)) (La version actuellement la plus diffusée étant la 5.1, vous trouverez dans des encadrés les différences par rapport à la version 6 actuellement disponible).

Une fois l'archive obtenue, décompilez-la dans le répertoire précédemment créé. Vous devez alors avoir un répertoire « **jboss-6.0.0.xxxxxxx-M4** » qui contient un répertoire bin. Dans ce répertoire bin, double cliquez sur le fichier run.bat sous Windows (sous Unix, lancez le run.sh).

Si tout se passe bien, vous ne devez pas avoir d'erreur dans les traces et la dernière ligne doit se finir par :

Started in 58s:376ms

Si ce n'est pas le cas, reportez-vous à la documentation très complète de JBoss.

Une fois cette étape franchie, il nous reste à dupliquer l'installation pour avoir un deuxième serveur JBoss.

Pour ce faire, arrêtez le serveur en cours (fermez la fenêtre de commande), puis recopiez entièrement le répertoire « **jboss-6.0.0.xxxxxxx-M4** » en un répertoire « **jboss-6.0.0.xxxxxxx-M4 -server2** ».

Si maintenant vous lancez les deux serveurs, vous allez avoir une erreur du style :

java.lang.Exception: Port 8083 already in use

Cette erreur est normale ! Effectivement, vous lancez en réalité deux serveurs avec exactement la même configuration. Il va nous falloir changer les ports d'écoute sur le deuxième serveur. Pour faciliter l'opération et éviter toute erreur nous ajouterons 100 à tous les numéros de port trouvés.

A cette fin dans « `jboss-6.0.0.xxxxxxx-M4-server2/server/deploy/` » supprimez tous les répertoires sauf « `jbossweb-standalone` ». Renommez « `jbossweb-standalone` » en « `default` ». Suite à cela cherchez le fichier « `jboss-6.0.0.xxxxxxx-M4-server2/server/default/conf/bindingservice.beans/META-INF/bindings-jboss-beans.xml` » et incrémentez tous les numéros de port de 100.

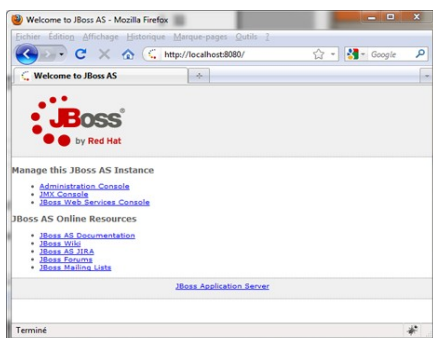
Relancez les deux serveurs et ouvrez un navigateur. Vous avez maintenant accès aux URLs :

`http://localhost:8080`

`http://localhost:8180`

En version 5.x de JBoss, il faudra garder et renommer le serveur « `web` » en « `default` », puis incrémenter tous les numéros de port dans le fichier « `jboss-5.1.0.GA-server2/server/default/deploy/jbossweb.sar/server.xml` »

Accédez, via la console JMX, à chacune des URLs précédentes, afin de vérifier le bon fonctionnement des serveurs.



Pour accéder aux consoles avec la version 5.X de JBoss utilisez le compte `admin/admin`.

Nous avons donc maintenant deux serveurs d'applications actifs. Il reste à mettre en place notre load balancer, dans notre cas un serveur web.

2.2. Installation d'Apache

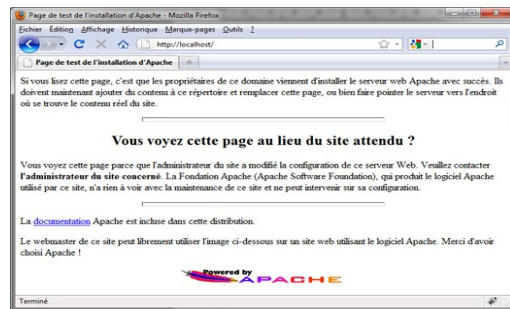
Apache s'avèrera plus simple à installer vu que nous n'avons besoin que d'un serveur de ce type. Pour des problèmes de compatibilité avec le module `mod_jk` que nous allons utiliser, je vous conseille de télécharger la dernière version disponible d'Apache 2.0 (dans mon cas la 2.0.63) sur le site d'Apache : [Lien49](#).

Exécutez l'installation et vérifiez en lançant le serveur que tout s'est bien passé.

Remarque 1 : La relance du serveur Apache se fait :

Dans votre navigateur, accédez alors à

l'URL `http://localhost/`, vous devez avoir un écran comme celui-ci :



Remarque2 : Attention sous Windows! Il vous faudra désactiver le serveur IIS qui utilise le port 80 sinon Apache ne pourra se lancer correctement. Pour ce faire allez dans « **menu démarrer>panneau de configuration>outils d'administration** » et lancez le « **gestionnaire de services internet** » ; là, vous aurez tout le loisir d'arrêter complètement IIS. Relancez à nouveau le serveur Apache et tout devrait être bon.

Nous avons maintenant trois serveurs indépendants et nous allons pouvoir entrer dans le cœur du sujet en commençant par une simple répartition de charge.

3. Configuration du load balancer avec le module `mod_jk`

Nous allons commencer par ajouter le module nécessaire à Apache afin de dialoguer avec les serveurs d'applications. Nous concernant, il existe plusieurs modules pour se connecter à JBoss :

- **mod_proxy** : module officiel livré avec Apache, il existe depuis 1996 et a été entièrement réécrit pour Apache 2.2 afin de concurrencer les autres modules ;
- **mod_cluster** : fourni par JBoss, c'est une solution jeune mais très prometteuse. Configurable dynamiquement, tenant compte de la charge des serveurs et capable de gérer l'arrêt des applications en douceur ;
- **mod_jk** : longtemps seul module professionnel, il est encore le module recommandé avec tomcat. Comme `mod_cluster`, il permet la modification à chaud par page web de la configuration.

Nous utiliserons `mod_jk` qui est le plus répandu. Pour ce faire, rendez-vous sur <http://tomcat.apache.org/download-connectors.cgi> ([Lien50](#)) et téléchargez la version binaire qui correspond à votre plateforme. Copiez le fichier « `mod_jk***.so` » dans le répertoire « `apache2/modules` » et renommez-le en « `mod_jk.so` ».

Ici nous ne toucherons pas aux serveurs d'applications ceux-ci étant considérés comme fonctionnels et à l'écoute de connexions avec le protocole AJP13.

Nous allons commencer par déclarer ces serveurs d'applications à notre load balancer (Apache 2). Pour ce faire rendez-vous dans « `apache2/conf` » et créez un fichier « `workers.properties` » dans lequel nous allons inscrire les serveurs de la manière suivante :

Workers.properties

```
# Define 1 real worker using ajp13
worker.list=loadbalancer,status
# Set properties for worker1 (ajp13)
worker.worker1.type=ajp13
worker.worker1.host=localhost
worker.worker1.port=8009
worker.worker1.lbfactor=1
worker.worker1.connection_pool_size=10
# Set properties for worker2 (ajp13)
worker.worker2.type=ajp13
worker.worker2.host=localhost
worker.worker2.port=8109
worker.worker2.lbfactor=1
worker.worker2.connection_pool_size=10
#fonctionnement de l'équilibrage de charge
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=worker1,worker2
worker.loadbalancer.sticky_session=True
worker.status.type=status
```

Si on analyse ce fichier, on peut voir plusieurs zones :

- une zone bleue qui nous permet de définir les workers qui seront visibles de l'extérieur. Ici le seul à nous intéresser est « **loadbalancer** » ;
- en vert et rouge, nous voyons la définition de connexion avec les deux serveurs d'applications. On voit que les deux connexions sont de type **AJP13** (standard de communication Apache-Tomcat). Les deux serveurs sont hébergés sur la machine d'où le localhost, par contre on peut voir que les ports de connexion sont différents. Ici nous avons réparti de manière égale les requêtes entre les serveurs car nous avons mis le même **lbfactor**. On voit aussi que l'on prévoit une dizaine de connexions par serveurs qui seront ouvertes au fur et à mesure ;
- en orange, enfin le vrai loadbalancer. On le reconnaît au « **type=lb** ». Ensuite on indique tous les serveurs à prendre en compte. Enfin la propriété « **sticky_session** » indique si le load balancing se fait avec affinité de session.

Remarque 3 : L'affinité de session indique que si une requête est retransmise sur le serveur 1 pour un client toutes les autres requêtes appartenant à la même session de ce client seront retransmises sur le serveur 1.

Il nous suffit maintenant de déclarer l'utilisation du module `mod_jk` à Apache ainsi que le fichier que nous venons de créer.

Pour cela nous allons modifier le fichier `httpd.conf` du même répertoire de la manière suivante :

Httpd.conf

```
...
LoadModule jk_module modules/mod_jk.so
...
# Where to find workers.properties
# Update this path to match your conf directory
location (put workers.properties next to
httpd.conf)
JkWorkersFile conf/workers.properties
# Where to put jk shared memory
```

```
# Update this path to match your local state
directory or logs directory
JkShmFile logs/mod_jk.shm
# Where to put jk logs
# Update this path to match your logs directory
location (put mod_jk.log next to access_log)
JkLogFile logs/mod_jk.log
# Set the jk log level [debug/error/info]
JkLogLevel info
# Select the timestamp log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "
JkMount /* loadbalancer
<IfModule worker.c>
StartServers2
MaxClients150
MinSpareThreads25
MaxSpareThreads75
ThreadsPerChild25
MaxRequestsPerChild0
</IfModule>
```

La ligne bleue permet d'ajouter `mod_jk` à la liste des modules chargés. En orange, nous indiquons le fichier de configuration à prendre en compte. En rouge, les lignes de configuration de sortie du module, à ne pas toucher.

La ligne verte nous intéresse plus, elle permet d'indiquer quelles URLs nous redirigeons et vers qui.

Ici toutes les requêtes sont redirigées sur le worker `loadbalancer`. Elles seront donc, au vu de notre configuration, réparties équitablement entre les deux serveurs.

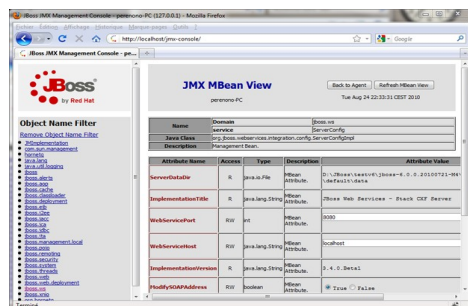
Il vous reste un dernier fichier à ajouter dans le répertoire `conf` pour faire un lien entre le worker `status` et une URL. Cette page `status` permettra d'afficher l'état du connecteur `mod_jk`. Ceci n'est en rien obligatoire mais conseillé en mode de développement/Pré-production.

Uriworkermap.properties

```
/status=status
```

À présent c'est le moment, relancez le serveur Apache pour prendre en compte cette nouvelle configuration (voir **remarque 1**).

Maintenant, vérifions le bon fonctionnement de notre configuration. Pour ce faire, ouvrez un navigateur et connectez-vous sur `http://localhost/`, vous devez maintenant voir la page d'accueil de JBoss et non plus celle d'Apache. Entrez sur la console JMX comme au début et cliquez sur « **JBoss.ws** » à gauche puis sur « **service=ServerConfig** ». Dans la fenêtre ainsi apparue vous voyez le port d'écoute du serveur : 8080 ou 8180.

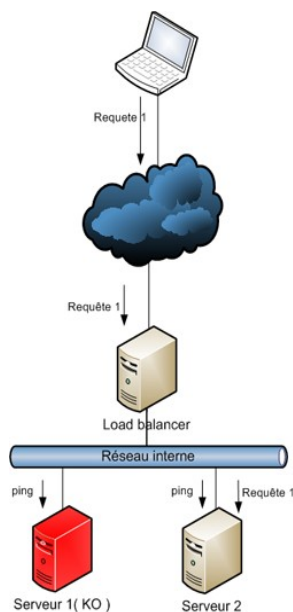


Ouvrez alors [une nouvelle fenêtre avec un autre navigateur](#) (si vous travaillez avec le même navigateur vous aurez la même session et donc accès au même serveur grâce à l'affinité de session). En reproduisant la même procédure, vous verrez que le serveur sera en écoute sur l'autre port.

4. Continuité de service

Maintenant que nous sommes capables de répartir la charge d'un site entre plusieurs machines, on peut se demander ce qui se passerait si un des serveurs venait à ne plus fonctionner. Et bien horreur rien du tout ! La communication étant rompue une erreur parviendra au client ce qui, vous en conviendrez, n'est pas du meilleur effet.

Pour répondre à cela tout a été prévu. On peut ainsi, au cas où Apache détecterait que le serveur auquel il tente d'accéder n'existe plus, lui dire de rediriger les requêtes vers un autre serveur.



Alors comment cela se passe-t-il ?

Et bien au moment où Apache reçoit une requête, il détermine le serveur qui doit répondre :

- soit une connexion libre est ouverte vers celui-ci et la requête est lancée ;
- soit aucune connexion n'est disponible, Apache va alors ping la machine pour savoir si elle existe ou non. Si le ping échoue, Apache regarde s'il existe une directive de redirection. Si c'est le cas, il va tenter de communiquer avec le serveur de remplacement.

Dans le cas qui nous intéresse, le serveur 1 étant en panne, Apache essaie d'ouvrir une connexion sur le serveur 2. Comme la connexion peut être créée alors la requête est émise vers ce nouveau serveur.

Pour ce faire, nous allons ajouter une ligne au worker1 dans notre fichier **workers.properties**.

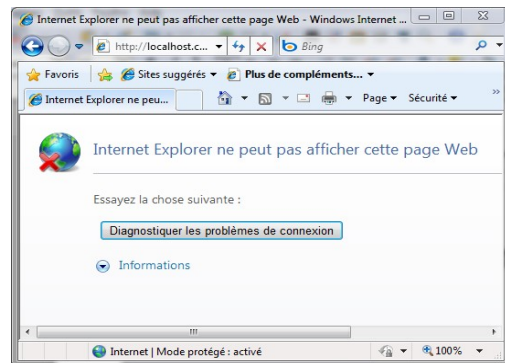
Workers.properties

```
...
# Set properties for worker1 (ajp13)
worker.worker1.type=ajp13
```

```
worker.worker1.host=localhost
worker.worker1.port=8009
worker.worker1.lbfactor=1
worker.worker1.connection_pool_size=10
# Define preferred failover node for worker1
worker.worker1.redirect=worker2
...
```

Nous venons d'indiquer que si le serveur worker1 n'est plus accessible les requêtes sont redirigées sur le worker2.

Pour tester cette configuration, arrêtez le serveur1 et retentez d'accéder depuis les deux navigateurs. L'un des deux navigateurs va afficher une erreur comme suit.



Relancez Apache pour qu'il prenne en compte la nouvelle configuration et refaites la manipulation. Maintenant, si vous vérifiez dans la console JMX, les deux navigateurs accèdent au même serveur.

Si nous nous contentons de cela, nous redirigeons toute la charge vers un serveur avec le risque qu'il ne puisse répondre à l'ensemble de la charge. Pour éviter ceci, on préfère ajouter un worker comme les autres mais que l'on met en attente. Ainsi, il ne recevra de requêtes que si l'un des serveurs tombe.

Workers.properties

```
...
# Set properties for worker2 (ajp13)
worker.worker3.type=ajp13
worker.worker3.host=localhost
worker.worker3.port=8209
worker.worker3.lbfactor=1
worker.worker3.connection_pool_size=10
# Disable worker3 for all requests except failover
worker.worker3.activation=disabled
...
```

Ajoutez le worker3 dans la propriété `worker.loadbalancer.balance_workers` pour le rendre disponible et bien se rappeler qu'un worker disabled est un worker qui ne reçoit pas de requête directement ; ainsi si vous définissez uniquement deux workers et que l'un est disabled toutes les requêtes iront sur l'autre serveur.

Encore quelques optimisations et nous voici avec un site prêt à répondre à la charge d'une situation réelle.

5. Optimisation annexe

Tout à l'heure nous avons vu une zone d'optimisation des connexions entre Apache et JBoss, il est temps de mieux comprendre à quoi elle sert.

```
Httpd.conf
...
<IfModule worker.c>
StartServers2
MaxClients150
MinSpareThreads25
MaxSpareThreads75
ThreadsPerChild25
MaxRequestsPerChild0
</IfModule>
```

Cette zone permet de définir le nombre de requêtes qui peuvent être acceptées en parallèle ainsi que le nombre de requêtes pouvant être mises en attente. Ainsi nous indiquons que deux processus serveurs sont créés au démarrage d'Apache, il n'influe que peu car Apache va au fur et à mesure des requêtes arrêter et lancer de nouveaux processus serveurs. On indique aussi que l'on pourra avoir au maximum 150 clients simultanément.

Remarque 4 : Il faut normalement compter 20Mo par client actif, on peut donc à partir de la mémoire libre définir la valeur de **MaxClients** à appliquer. Ainsi pour 1Go, on pourra accepter 50 clients en parallèle.

Les valeurs **MinSpareThreads** et **MaxSpareThreads** permettent d'entretenir des processus serveur de secours en fonction du nombre de requêtes.

Remarque 5 : Les valeurs **MinSpareThreads** et **MaxSpareThreads** doivent être en cohérence avec les valeurs présentes sur les serveurs d'applications.

ThreadsPerChild définit le nombre de threads au sein de chaque processus enfant.

La propriété **MaxRequestPerChild** est ici désactivée, elle permet de limiter le nombre de requêtes acceptées par processus serveur. Si ce maximum est atteint le processus serveur est cloné puis arrêté.

6. Conclusion

Il ne reste plus qu'à appliquer ce que nous avons vu sur un projet concret et se frotter au réglage plus fin du serveur. Mais là, je ne me fais pas de soucis la documentation existe en nombre, particulièrement pour JBoss et Apache.

Retrouvez l'article de Noël Perez en ligne : [Lien51](#)

ActionScript Facile - Chapitre 1 - Pourquoi créer des composants graphiques ?

Article original : ActionScript-Facile.com - Pourquoi créer des composants graphiques ? ([Lien52](#))

1. Introduction

Pourquoi créer des composants graphiques ? Après tout, ce qui est graphique est censé revenir de droit aux graphistes non ? Alors pourquoi un développeur irait s'ennuyer avec cela ?

En vérité, il existe une multitude de bonnes raisons et je vais vous montrer l'utilité de **développer des composants graphiques génériques et réutilisables**.

Pour commencer, nous allons aborder le thème de la réutilisation du code source, chose qui devient essentielle dès que l'on commence à vouloir développer des applications avec interfaces utilisateur.

2. Réutilisation du code source

Avant tout et pour commencer nous allons définir clairement ce qu'est un composant graphique.

Un composant graphique est un morceau de code source qui va nous permettre d'offrir à l'utilisateur une information ou une interaction sous forme graphique. Cela peut aller de l'infobulle à la barre de défilement (scrollbar). Ce code source se veut réutilisable et en général skinnable, c'est à dire que l'on peut personnaliser son apparence et donc changer le côté graphique autant de fois qu'on le désire.

Maintenant que nous avons fixé les limites de ce qu'est un composant graphique, nous pouvons nous intéresser à ce qui fait qu'ils sont si utiles.

Pour cela nous allons nous figurer une scène classique de début de projet : Tom est développeur, il doit mettre en place un formulaire permettant à ses utilisateurs de créer leur CV en ligne. Son chef de projet vient le voir et lui demande de commencer le développement de l'application tout de suite. Le problème c'est qu'aucune charte graphique n'a été fournie et que le code qui permet d'enregistrer les CV en base de données est déjà prêt. Il se demande donc à juste titre ce qu'il va pouvoir faire car il a pris l'habitude de reprendre les chartes des graphistes et développer ses fonctionnalités en mélangeant le fond et la forme.

Voilà un problème que rencontre tout programmeur débutant, en effet, il n'est pas naturel de dissocier les éléments graphiques du code. Cela demande une certaine pratique de raisonnement dans l'abstrait, or raisonner dans l'abstrait c'est comme tout, ça se travaille.

Voyons comment Tom peut se sortir de ce pétrin et

avancer le développement sans charte graphique et ce, de manière intelligente.

Il a obtenu une liste des fonctionnalités de ce formulaire, il prend donc une feuille et un crayon et remarque qu'il va devoir coder au moins trois ou quatre boutons, d'aspects différents certes, mais **les fonctionnalités resteront les mêmes**.

Il se rend compte également que toutes les informations doivent tenir sur une page mais que si l'on est un peu pointilleux, les champs de texte libre peuvent prendre énormément de place. Il va donc falloir masquer le contenu qui sort des limites et le faire défiler à l'aide d'une barre.

Tom va donc coder les fonctionnalités, comme le défilement et les différents états d'un bouton, toutefois son code ne devra dépendre d'aucun graphisme tout en laissant une possibilité de personnaliser l'apparence du composant. Ainsi, quelle que soit la charte graphique qu'on lui fournira, il aura juste à l'intégrer sur chaque bouton et sur chaque scrollbar.

Voici donc **le premier intérêt des composants graphiques, la possibilité de les réutiliser**. Ils permettent de gagner du temps car une fois créés, il n'y a plus matière à revenir dessus. Cela demande du temps et de l'expérience pour créer des composants efficaces réutilisables partout.

En général, **les composants sont efficaces** lorsque vous arrivez à **les utiliser sur trois ou quatre projets différents sans aucune retouche** à réaliser.

Nous allons maintenant aborder brièvement **la deuxième bonne raison de créer des composants graphiques** qui découle inévitablement de la première, **la facilité de développement des applications avec interfaces utilisateur**.

3. Le développement d'interfaces utilisateur à l'aide des composants

Les interfaces utilisateur sont parmi les choses les plus pénibles à réaliser pour un développeur et ce pour plusieurs raisons en voici une liste non exhaustive :

- en général il nous est souvent demandé de changer telle ou telle chose à la dernière minute et ce, de façon répétée ;
- le rendu fonctionnel, de manière générale, n'est pas aussi bon que ce à quoi l'on s'attendait. Ce qui entraîne des ajustements de bouts de ficelle. Nous

avons tous vécu la scène du graphiste qui vient se poser à côté de vous « pour 5mn » et qui repart une heure après, tout content d'avoir bataillé sur chaque pixel. Notez cependant que cela ne l'empêchera pas de reprendre entièrement le concept visuel et ce, dès le lendemain ;

- il y a clairement un manque de défi technique dans les interfaces utilisateur, en général quand on en a fait deux ou trois, on les a toutes faites et les suivantes nous paraissent insipides. Raison de plus pour y passer le moins de temps possible et occuper son esprit avec des choses plus motivantes.

Nous avons déjà vu qu'une fois **correctement développés, les composants graphiques peuvent être réutilisés à l'infini et personnalisés à souhait**. Ce qui résulte en **un gain de temps considérable dans le développement d'une application**.

Pour reprendre l'exemple de Tom, admettons qu'il ait déjà créé ses composants "scrollbar" et "boutons" lors d'un projet précédent, il pourra raisonnablement répondre à son responsable : « écoute, j'ai déjà une liste de composants graphiques prêts à l'emploi, je vais plutôt passer du temps sur autre chose ».

Et voilà, **Tom a gagné du temps**, son responsable est content et lui aussi.

4. Conclusion

En conclusion, nous pouvons affirmer qu'il y a énormément **d'avantages à créer des composants graphiques**, et ce, quel que soit le type de projet pour lequel nous sommes amenés à travailler.

Nous avons vu que **les composants représentent une manière élégante et astucieuse de développer des interfaces utilisateur**, et que, de par leur **nature réutilisable**, ils permettent de **gagner du temps de développement considérable**.

Dans les prochains chapitres, nous verrons ensemble comment créer des composants et comment penser leur organisation.

Ensuite, nous enchaînerons avec une liste d'articles à thème, chaque article traitant de la création d'un composant spécifique, et avec le code source bien évidemment.

A bientôt.

Retrouvez l'article de Matthieu Deloison en ligne : [Lien53](#)

JavaScript Orienté Objet : syntaxe de base des classes JavaScript à l'intention des développeurs PHP

Ce tutoriel a pour cible les développeurs qui ont une expérience du PHP (5) et qui veulent se lancer dans un projet JavaScript qui dépasse le simple scripting. Cela va donc commencer par savoir écrire des classes en JavaScript. Pour avoir galéré en tant que développeur puis en tant que lead technique, avoir formé de bons développeurs PHP à faire des applications Web où le JavaScript représente plus d'un tiers du code et la moitié du temps de développement, j'ai pu constater les énervements classiques lorsqu'on commence à vouloir faire des choses sérieuses en JavaScript.

Le but ici n'est pas de rentrer dans la théorie du langage JavaScript ou même d'être exhaustif (voir à la fin de cet article des ressources qui le font très bien), mais de vous fournir un template pour commencer à écrire vos classes.

1. JavaScript c'est compliqué

Quand on arrive du PHP, du C ou même de Java, JavaScript peut être franchement surprenant. Certains s'en amusent ([Lien54](#)), d'autres prennent sa défense en rappelant son histoire mouvementée (la fusion de trois langages, une implémentation en quelques semaines, pris dans la Browser War depuis 15 ans) ([Lien55](#)) et surtout une chose qui est bien particulière aux développeurs Web : **personne ne prend la peine de l'apprendre !**

Ajouté à cela, il y a le **DOM** dont l'implémentation dans chaque browser varie, la **programmation événementielle** que les développeurs PHP n'ont en général jamais expérimentée, le manque de **documentation centralisée** (pas d'équivalent à PHP.net) et enfin la version implémentée d'ECMAScript qui varie selon le navigateur (pour info, il faut en rester à la **version 1.5** qui est celle de IE6-8).

Concrètement, il y a deux choses à comprendre pour éviter les erreurs classiques et partir sur une bonne base de code pour programmer avec des objets :

- **le scope** : repérer et utiliser `var` et *les closures ou `}`* qui l'entourent, vos variables ne sont visibles qu'à l'intérieur ;
- **tout est fonction()** : fonctions, méthodes, classes, constructeurs sont créés de cette seule manière.

2. Architecture comparée JavaScript/PHP

2.1. Éviter les globales, utiliser `var` et les namespaces

Valable dans les deux langages : vous allez essayer de minimiser le nombre de variables disponibles dans `$_GLOBALS` et `window.*` et donc modifiables par les librairies que vous incluez, par l'arrivée d'un nouveau code ou par toute modification de l'existant.

Exemple, ceci génère une boucle infinie :

```
function genericFunctionName() {
for(i = 0; i < myArray.length; i++) {
    ....
}
```

```
for(i = 0; i < 10; i++) {
    genericFunctionName();
}
```

On a créé ici une **boucle infinie** car le `i` à l'intérieur et à l'extérieur de la fonction fait référence à `window.i` : c'est la même variable globale. Ici la première librairie venue (ou publicité) risque en outre **d'écraser le nom de votre fonction** s'il est trop générique : j'ai déjà vu par exemple *jQuery* et *l'API Mappy* se gêner l'un l'autre sur la même page ! Et, même si JavaScript est monothread, il y a des cas où la valeur de `i` peut être modifiée par une autre boucle qui utilise ce nom si répandu.

La solution ici est de rajouter `var` pour que le `i` à l'intérieur de la fonction ne soit pas visible de l'extérieur. **Son scope est la closure la plus proche**, c'est-à-dire la déclaration de fonction la plus proche.

```
for(var i = 0; i < myArray.length; i++)
```

Pour partir sur de bonnes bases, on va **créer un namespace** pour tout le code de notre application Web. Pendant tous les développements, il faudra prendre l'habitude de déclarer ses variables avec `var`.

```
var MY_APP_NAMESPACE = {}; // l'équivalent
namespace de PHP5.3 n'existe pas, on déclare un
objet JavaScript
```

```
MY_APP_NAMESPACE.genericFunctionName = function()
{
    var aMyArray = [ ... ], // multiples
déclarations de variables locales
    iTotale = aMyArray.length;
    for(var i = 0; i < iTotale; i++) ....
}; // notez le ; final, on a tendance à l'oublier

for(i = 0; i < 10; i++) {
    MY_APP_NAMESPACE.genericFunctionName();
}
```

La boucle dans la fonction est protégée, ni `i` ni le tableau n'est accessibles de l'extérieur. Il est peu probable que des codes extérieurs utilisent votre *namespace* et s'ils le font,

vous le sentirez de toute manière passer, ce qui (sans ironie) est plus facile à détecter qu'un petit bogue !

3. Classes statiques

Toujours dans l'idée de **libérer notre espace global**, c'est une bonne pratique en PHP comme en JavaScript d'arrêter d'accumuler des listes sans fin de fonctions : il vaut mieux les **regrouper par thème** dans des « **classes statiques** » en PHP5, dans des **sous-namespace en PHP5.3-JavaScript**.

Exemple d'une classe **PHP** servant à valider le format d'*input* utilisateur :

```
namespace MY_APP_NAMESPACE;
class validation {
    public static $regPassword='^[a-zA-Z0-9.\|\/\+=%ùf*^`_&!@#-]{3,50}$';

    static function isValidMail($sMail) {
        return (filter_var($sMail,
        FILTER_VALIDATE_EMAIL));
    }
    static function isValidPassword($sPassword) {
        return (preg_match("/".self::$regPassword."/",$sPassword) === 1);
    }
}
```

De n'importe où, en PHP, on peut donc appeler ces fonctions de cette manière :

```
print
MY_APP_NAMESPACE\validation::isValidMail( 'mon@ma
il.com' );
```

Voici l'équivalent JavaScript ([Lien56](#)) :

```
MY_APP_NAMESPACE = {};

(function(){ // début de scope local
MY_APP_NAMESPACE.utils = MY_APP_NAMESPACE.utils
|| {};
// déclaration de la classe de validation
proprement dite
MY_APP_NAMESPACE.utils.validation = {
    // déclaration de nos variables statiques
    regMail: /^[w-]+(?:.[w-]+)*(?:[w-]+\.)+
[a-zA-Z]{2,7}/,
    regPassword: /^[a-zA-Z0-9.\|\/\+=%ùf*^`_&!@#-]
{3,50}/,
    // déclaration de nos méthodes
    isValidMail:function( sMail ) {
        return (self.regMail.exec( sMail ) !=
null);
    },
    isValidPassword:function( sPassword ) {
        return
(self.regPassword.exec( sPassword ) != null);
    }
}; // fin de classe

// trick JavaScript pour émuler le self:: en
PHP : on utilise une variable locale
var self = MY_APP_NAMESPACE.utils.validation;
})(); // fin de scope local
```

De n'importe où, en JavaScript, on peut donc appeler ces fonctions de cette manière :

```
alert(MY_APP_NAMESPACE.utils.validation.isValidMail(
'id( 'monm@il.com' )); //true
alert(MY_APP_NAMESPACE.utils.validation.isValidMail(
'id( 'moççna@il.com' )); // false
```

La syntaxe est radicalement différente de PHP car **il n'y a rien d'explicite** et on exploite plusieurs spécificités JavaScript, mais on est arrivé au même résultat. Je vous conseille de prendre ce bout de code comme un template pour créer des classes statiques JavaScript.

Si vous aimez comprendre :

- la première et la dernière ligne sont des fonctions anonymes auto-exécutées que l'on met autour de chaque classe. Elles servent à avoir un scope local propre à la classe et donc à **émuler des variables privées pour cette classe** ;
- la deuxième ligne suppose que MY_APP_NAMESPACE a déjà été déclarée mais n'est pas certaine du sous-namespace utils. La notation spéciale || (double pipe) permet donc de créer ce sous-namespace uniquement s'il n'est pas déjà déclaré (et donc de ne pas l'écraser) ;
- la classe statique validation est **concrètement un objet JavaScript** composé de deux expressions régulières (directement compilées avec / /) et de deux fonctions. Le tout est déclaré avec la notation **JSON** : { clé : valeur , ... }. Attention à ne jamais laisser traîner une virgule derrière la dernière clé, car cela fait planter IE et il ne le dit pas clairement ;
- dans l'avant-dernière ligne, on déclare une variable privée qu'on appelle self et qui remplit la même fonction qu'en PHP : faire référence à notre classe statique. Comme en PHP elle est là pour des raisons de confort (éviter de retaper entièrement et en dur le nom de la classe).

4. Objets instanciables

Créons une classe PHP classique avec constructeur, méthode publique, variable privée et variable statique publique. Prenons par exemple un objet permettant de manipuler des dates :

```
namespace MY_APP_NAMESPACE;
class customDate {
    private $iTimestamp = 0; // variable privée
    propre à chaque instance
    static $aMonthNames =
    array('January', ...); // variable statique
    partagée pour tout le code
    // constructeur
    public function __construct($iTimestamp) {
        $this->iTimestamp = $iTimestamp;
    }
    // méthode publique propre à chaque instance
    public function getMonthName() {
        $month_number = date('n', $this->iTimestamp)
-1;
        return self::$aMonthNames[$month_number];
    }
}
```

Utilisation :

```
$date = new
MY_APP_NAMESPACE\customDate( 1268733478547 );
print $date->getMonthName(); // 'March'
```

Maintenant en JavaScript ([Lien57](#)) :

```
(function(){ // début de scope local
MY_APP_NAMESPACE.utils = MY_APP_NAMESPACE.utils
|| {}}; // création d'un sous namespace pour y
stocker nos classes utilitaires si celui-ci n'est
pas déjà créé

// constructeur
MY_APP_NAMESPACE.utils.customDate =
function( iTimeStamp ) {
    this.iTimeStamp = iTimeStamp;
    this.date = new Date();
    this.date.setTime(this.iTimeStamp);
};

// variables et méthodes publiques propres à
chaque instance
MY_APP_NAMESPACE.utils.customDate.prototype = {
    date:null,
    iTimeStamp:0,
    getMonthName:function() {
        var iMonthNumber =
this.date.getMonth();
        return self.aMonthNames[iMonthNumber];
    }
};

// variable statique partagée pour tout le code
MY_APP_NAMESPACE.utils.customDate.aMonthNames =
['January', 'February', 'March', 'April', 'May',
'June', 'July', 'August', 'September', 'October',
'November', 'December'];

// trick JavaScript pour émuler le self:: en
PHP : on utilise une variable locale
var self = MY_APP_NAMESPACE.utils.customDate,
privateVariable = 0; // variable privée visible
par toutes les instances

})(); // fin de scope local
```

Remarquez que cette syntaxe (dite *prototype*) n'autorise pas à avoir des variables privées pour chaque instance comme en PHP: ici `this.iTimeStamp` se réfère bien à l'instance de `customDate`, mais est accessible depuis l'extérieur. La variable vraiment privée est `privateVariable` mais elle est visible et modifiable par toutes les instances, concept qui serait équivalent en PHP à une **variable statique privée**, qui peut par exemple servir à un manager d'instance (pattern factory + accessor) pour stocker une liste des instances en cours.

4.1. Syntaxe alternative

Il existe une autre syntaxe, dite *closure*, qui permet d'avoir des variables et méthodes privées pour chaque instance mais elle est moins performante si vous instanciez des centaines d'objets, ou que vos objets commencent à contenir plusieurs méthodes (voir ce petit bench [Lien58](#)), il en existe des dizaines d'autres qui confirmeront la même chose). À vous de faire la balance entre avoir des variables privées et de meilleures performances ([Lien59](#)) :

```
MY_APP_NAMESPACE = {};

(function(){ // début de scope local
MY_APP_NAMESPACE.utils = MY_APP_NAMESPACE.utils
|| {}}; // création d'un sous namespace pour y
stocker nos classes utilitaires si celui-ci n'est
pas déjà créé

// constructeur
MY_APP_NAMESPACE.utils.customDate =
function( iTimeStamp ) {
    // ces variables resteront privées,
spécifiques à l'instance
    var iTimeStamp = iTimeStamp,
        date = new Date();
    date.setTime(iTimeStamp);

    // on renvoie ce qui est public sous la forme
d'un objet
    return {
        getMonthName:function() {
            var iMonthNumber = date.getMonth();

            return
self.aMonthNames[iMonthNumber];
        }
    };
};

// variable statique partagée pour tout le code
MY_APP_NAMESPACE.utils.customDate.aMonthNames =
['January', 'February', 'March', 'April', 'May',
'June', 'July', 'August', 'September', 'October',
'November', 'December'];

// trick JavaScript pour émuler le self:: en
PHP : on utilise une variable locale
var self = MY_APP_NAMESPACE.utils.customDate,
privateVariable = 0; // variable privée
visible par toutes les instances
})(); // fin de scope local

// privateVariable est privé
console.log('variable privée :'+ typeof
privateVariable ); // undefined
// création d'un objet date
var date1 = new
MY_APP_NAMESPACE.utils.customDate(1268733478547);
// 16 mars 2010, 10:58
console.log('méthode publique d
instance :'+date1.getMonthName()); // March
console.log('variable privée d instance :'+
typeof date1.date ); // undefined

// la variable statique est disponible en lecture
console.log('variable statique :
'+MY_APP_NAMESPACE.utils.customDate.aMonthNames[0
]); // January

// et aussi en écriture, ici on change de langue
à la volée
MY_APP_NAMESPACE.utils.customDate.aMonthNames =
['Janvier', 'Février', 'Mars', 'Avril', 'Mai',
'Juin', 'Juillet', 'Août', 'Septembre',
'Octobre', 'Novembre', 'Decembre'];
console.log( date1.getMonthName() ); // Mars
```

Ici on a remplacé l'utilisation de prototype pour ajouter des propriétés par un return dans le constructeur d'un objet contenant des propriétés publiques. Comme on se trouve dans le scope du constructeur, les méthodes ont accès à `iTimeStamp` et `date` qui sont **privées et propres à**

chaque instance. Le coût en performances vient du fait que les fonctions sont redéfinies à chaque fois que l'on crée un objet, alors qu'avec *prototype*, la définition n'était faite qu'une seule fois.

5. Conclusion

Vous voici donc avec la notion salutaire de *namespace*, une implémentation de *self::*, une idée sur le fonctionnement du scope et **deux templates de classes de base** (en mode *prototype* et en mode *closure*, le premier étant à préférer). Ces templates m'ont servi ces quatre dernières années sur des projets JavaScript d'envergure moyenne (+ de 100 classes, des milliers de lignes de code), ce modèle est donc éprouvé.

Notez que je n'ai **pas traité l'héritage des classes**, c'est parce que je suppose que si vous en êtes à vouloir développer en JavaScript Orienté Objet, vous êtes probablement sur un projet suffisamment large pour utiliser des bibliothèques JavaScript comme YUI ([Lien60](#)) ou jQuery ([Lien61](#)) qui ont chacune des méthodes pour simuler l'héritage (qui n'existe pas formellement en JavaScript). Maintenant si ça vous intéresse, je peux faire un petit post là-dessus, **dites-moi ça dans les commentaires**.

Retrouvez l'article de Jean-Pierre Vincent en ligne : [Lien62](#)

Comment créer facilement un framework Javascript - Partie 4

Traduction de l'article How to Easily Create a JavaScript Framework, Part 4 de Teylor Feliz paru sur AdmixWeb ([Lien63](#)).

Translated with the permission of AdmixWeb and the author.

1. Introduction

Dans Comment créer facilement un framework Javascript - Partie 3 ([Lien64](#)), nous avons découvert de nouvelles techniques : "Attendre jusqu'au chargement du DOM", "Modification de la fonction 'setStyle()", "Méthodes set et get pour les éléments d'entrée" et "Effets amusants : 'fadeIn()' et 'fadeOut()'". Cette semaine, je vais terminer la série de tutoriels sur le Framework Javascript "VOZ" en ajoutant de nouvelles méthodes utiles dans un framework Javascript. En bonus, je vais utiliser un peu d'AJAX dans ce framework. N'hésitez pas à utiliser ce framework tel quel ou en ajoutant vos propres méthodes. J'espère que vous trouverez ce dernier article de la série aussi intéressant que les trois premiers ! N'hésitez pas à laisser vos commentaires, et amusez-vous !

2. Supprimer des événements avec la fonction "UN"

Ici, nous allons faire l'inverse de la méthode "ON" qui est utilisée pour ajouter des événements aux éléments. Précédemment, nous ne pouvions pas supprimer l'évènement que nous avons ajouté sur un élément, maintenant si. Par exemple :

```
// Supprime des événements sur les éléments
// Par exemple $
$.getById('elem').un('click', alertme);
un:function(action, callback){
    // Vérifie si la méthode
    removeEventListener est disponible
    // Ceci fonctionne pour tous les
    principaux navigateurs sauf IE
    if(this.elems[0].removeEventListener){
        for(var i = 0; i < this.elems.length; i++){
            // Suppression de l'évènement sur les
            éléments
            this.elems[i].removeEventListener(action,
            callback, false);
        }
    }
    // Si le navigateur est IE :( nous utilisons la
    méthode detachEvent
    else
    {
```

```
for(var i = 0; i < this.elems.length; i++){
    // Suppression de l'évènement sur
    les éléments, uniquement pour IE
    this.elems[i].detachEvent('on'+action, ca
    llback);
}
return this;
}
```

3. Méthode \$\$.getByInstance()

Avec cette méthode, nous avons juste besoin d'enregistrer l'Objet de l'élément à l'intérieur du framework, pour pouvoir l'utiliser. Comme pour les autres méthodes, la description se trouve directement dans le code. Par exemple :

```
// Récupère les éléments par instance
// Par exemple, si vous faites référence au corps
du document, vous devrez utiliser "document.body"
au lieu de "document"
getByInstance:function(elem){
    var elems = [];
    elems.push(elem);
    this.elems = elems;
    return this;
}
```

4. Méthodes innerHTML

Cette méthode fait la même chose que la méthode JavaScript innerHTML, avec comme particularité de pouvoir ajouter du code HTML dans l'élément trouvé. Par exemple :

```
// Méthode innerHTML pour insérer du code HTML à
un élément
// html est le paramètre contenant le code html à
insérer dans l'élément
innerHTML:function(html){
    for(var i = 0; i < this.elems.length; i++){
        this.elems[i].innerHTML= html;
    }
    return this;
}
```

5. Méthodes text

Cette méthode est une autre façon d'insérer du texte dans le DOM. Cependant, faites attention car cela remplacera tout le précédent contenu de l'élément. Dans mon travail, je trouve cette méthode très utile. Voici un exemple :

```
// Insertion de texte dans un élément
// Remplacement du précédent contenu
text:function(text){
    text = document.createTextNode(text);
    for(var i = 0; i < this.elems.length;i++){
        this.elems[i].innerHTML = '';
        this.elems[i].appendChild(text);
    }
    return this;
}
```

6. Objet AJAX

Nous y voilà. Commençons par créer un objet pour pouvoir utiliser les requêtes GET/POST en AJAX. L'objet a seulement deux méthodes. La première, "getxhr()", est utilisée pour obtenir un XMLHttpRequest sans se préoccuper du navigateur. Pour Internet Explorer, nous nous limiterons aux versions 6 ou supérieures. L'autre méthode est "sendRequest()", qui est celle que nous utiliserons pour créer une requête AJAX. Créons l'objet AJAX à l'intérieur du framework "VOZ" de façon à l'utiliser comme cela : "\$\$.AJAX.method." Vous pouvez ainsi récupérer l'objet AJAX et l'utiliser en dehors du framework pour une autre utilisation, cela ne l'empêchera pas de fonctionner normalement.

```
AJAX: {}
```

a. Méthode "getxhr"

Cette méthode est créée spécialement pour les développeurs qui ne veulent pas utiliser de framework AJAX et préfèrent créer leurs propres requêtes. Cette méthode renverra simplement l'objet XMLHttpRequest. Par exemple :

```
// Cette méthode crée un objet XHR
// Si vous voulez créer vos propres requêtes
AJAX, utilisez cette fonction
// Par ex. : var myXHR = $$AJAX.getxhr();
getxhr:function(){
    var xhr = (window.XMLHttpRequest)
    ? new XMLHttpRequest() : new
    ActiveXObject("Microsoft.XMLHTTP");
    return xhr;
}
```

Maintenant nous disposons d'une méthode pour créer un objet xhr multinavigateur. Utilisons-la dans la méthode "sendRequest".

b. Méthode "sendRequest"

La méthode "sendRequest" peut être utilisée pour les deux types de requête : GET et POST. Nous devons juste le préciser dans un des paramètres. Par exemple :

```
// Paramètres de sendRequest(method, url,
ValuesToSend, callbackObject)
```

L'utilisation de cet objet est assez facile :

```
$$AJAX.sendRequest('post','contact.php',
{nom:'Al',email:'pacino@mail.com', message:'Hello
world'},
{success:function(x)
(alert(x.responseText))});
```

Le dernier argument est un objet contenant trois méthodes différentes :

- **Success**
lorsque la requête est satisfaisante ;
- **Loading**
fonctionne après l'envoi de la requête (du client) et avant la réception de la réponse (du serveur) ;
- **Error**
s'il y a une erreur de n'importe quel type lors de l'envoi de la requête.

```
// Méthode principale AJAX
// Paramètres de sendRequest(method, url,
ValuesToSend, callbackObject)
// Exemple : $
$.AJAX.sendRequest('get','client.php',
{clientId:1234, prenom:'James', nom:'Bond'},
//
{success:function(){alert('oui')},
loading:function(){alert('Attendez SVP')},
//
error:function()
{alert('Non')}});
sendRequest:function(m,url,valObj,callObj){
    // Nous créons ici l'objet xhr
    var myxhr = this.getxhr();
    // Puisque nous devons envoyer les
valeurs dans une chaîne de caractères.
    // Exemple : monurl.php?
parametre1=valeur1&parametre2=valeur2...
    // Nous allons créer la variable 'values'
qui sauvegardera celles-ci
    // En parcourant l'objet valObj nous
récupérons son nom et sa valeur
    // Enfin nous utilisons
encodeURIComponent pour échapper les caractères
spéciaux
    var values = '?';
    for(var k in valObj){
        values+= encodeURIComponent(k) +
        '=' + encodeURIComponent(valObj[k]) + '&';
    }
    // Si la méthode choisie est 'GET', nous
ajoutons les valeurs à l'URL
    // Comme nous n'avons pas besoin
d'envoyer les valeurs par le xhr.send, nous
mettons à null la variable 'values'
    if(m === 'get'){
        url+=values;
        values= null;
    }
    // Ouverture de la connexion AJAX
myxhr.open(m,url,true);

    // Si la méthode est 'POST'
    // Nous devons supprimer le '?' au début
de la valeur de la variable 'values'
    // Et définir l'en-tête requis pour la
méthode 'POST'
    if(m==='post'){
        values=values.substring(1,values.
length-1);
```

```

        myxhr.setRequestHeader("Content-
Type", "application/x-www-form-urlencoded");
    }
    // Envoi de la requête avec AJAX
    myxhr.send(values);
    // Si elle est disponible, exécute la
fonction de chargement du callObj
    if(callObj.loading){ callObj.loading();}
    // Lorsque 'readyState' vaut 4
    myxhr.onreadystatechange = function(){
        if(myxhr.readyState==4){
            switch(myxhr.status){
                // Si le status
vaut 200 cela veut dire que tout s'est bien
passé,
                // on appelle la
fonction success de callObj.
                case 200:
                    if(callObj.success)callObj.success(myxhr); break;
                // Si le status
vaut 403, 404 ou 503, cela signifie qu'il y a eu
un problème,
                // on appelle la
fonction error de callObj.
                case 403, 404,
503 : if(callObj.error)callObj.error(myxhr);
break;
                default:

```

```

callObj.error(myxhr);
            }
        }
    }
}

```

7. Conclusion

Exemple du framework JavaScript VOZ partie 4 : [Lien65](#)

Fichier js VOZ partie 4 : [Lien66](#)

Visitez le lien ci-dessus pour voir la dernière partie de notre framework VOZ en action ! Aussi, restez connecté pour de futurs tutoriels Javascript et AJAX sur lesquels je travaille actuellement et, qui devraient arriver dans les prochains mois. [Notes de traduction : les autres tutoriels de l'auteur sont disponibles sur son site admixweb ([Lien67](#))]

Comme toujours, j'espère que vous avez trouvé ce tutoriel amusant et facile à suivre ! N'hésitez pas à laisser vos commentaires, car comme toujours j'apprécie l'avis des amoureux du Web !

Retrouvez l'article de Teylor Feliz traduit par KalyParker en ligne : [Lien68](#)



Les dernières news

Sortie de Boost 1.45

Bonjour,
La nouvelle mouture de Boost vient de sortir. Nous en sommes donc à la 1.45.

Pour télécharger cette version : [Lien69](#)

Les nouveautés : essentiellement des corrections dans beaucoup de modules.

N'hésitez pas à donner ici vos impressions sur cette version de boost :

- => Utilisez-vous Boost ?
- => Quelles bibliothèques font partie de tous vos projets ?
- => Quelles sont vos bibliothèques préférées ?
- => Quelles bibliothèques supplémentaires souhaiteriez vous voir ?

***** Qu'est ce que Boost ?

Boost est une collection de bibliothèques génériques écrites en C++. Boost est gratuit et peut être utilisé avec tout type de programme (gratuit, commercial, opensource, sources fermées, etc.).

Boost vous permet ainsi de développer des programmes sans avoir à réinventer la roue en proposant des solutions très ouvertes à des besoins courants.

***** Que trouve-t-on dans Boost ?

En vrac et de façon non exhaustive, Boost propose des bibliothèques pour :

- Boost.Asio : bibliothèque réseau générique ;
- Boost.Bind : généralisation des `std::bind1st` et autres en attendant leur intégration dans C++0x ;
- Boost.FileSystem : gestion générique des systèmes de fichiers ;
- Boost.Interprocess : pour la communication entre les processus ;
- Boost.Iterator : d'intéressants itérateurs et de quoi construire facilement les vôtres ;
- Boost.Python : interfacer vos programmes C++ avec Python ;
- Boost.Random : nombres aléatoires ;
- Boost.Regex : gestion des expressions régulières ;
- Boost.Signal : pas besoin de déléguer ou de

précompilation et autres moc pour implémenter des signaux et des slots ;
Boost.Smart Pointer : pointeurs intelligents pour ne plus avoir de problème de mémoire ;
Boost.Thread : multithreading générique et facile ;
Boost.TR1 : pour les compilateurs n'ayant pas encore TR1.
Beaucoup de bibliothèques pour la programmation générique :
de nouveaux algorithmes ou conteneurs ;
de quoi gérer des graphes ou des automates ;
des Mathématiques ;
etc.

***** C'est si bien que ça Boost ?

Cette collection de bibliothèques propose une approche moderne pour le développement C++. Le code est ouvert et fait l'objet de revue de la part de la communauté pour assurer un code de qualité et ouvert. Beaucoup de ces propositions seront intégrées dans la future norme C++0x. Mais laissons parler les grands noms :

"...one of the most highly regarded and expertly designed C++ library projects in the world."

Herb Sutter and Andrei Alexandrescu, C++ Coding Standards ([Lien70](#))

"Item 55: Familiarize yourself with Boost."

Scott Meyers, Effective C++, 3rd Ed ([Lien71](#)).

"The obvious solution for most programmers is to use a library that provides an elegant and efficient platform independent to needed services. Examples are BOOST..."
Bjarne Stroustrup, Abstraction, libraries, and efficiency in C++ ([Lien72](#))

***** Boost sur developpez.com ?

=> F.A.Q. : Questions sur Boost ([Lien73](#))

=> Articles : Tutoriels sur des bibliothèques Boost ([Lien74](#))

=> Livres : C++ Template Metaprogramming : Concepts, Tools, and Techniques from Boost and Beyond de David Abrahams et Aleksey Gurtovoy ([Lien75](#))

Commentez cette news en ligne : [Lien76](#)



La traduction en français de Qt Creator 2.1 beta 2, une réalisation de la rubrique Qt de Developpez.com

La sortie de Qt Creator 2.1 Beta 2 a été annoncée aujourd'hui sur les labs blogs ([Lien77](#)).

Une fois de plus, l'équipe de la rubrique Qt de Developpez.com s'est donné du mal pour vous fournir une version française de cette version avec la traduction de plus de 1000 chaînes de caractères supplémentaires depuis la version précédente, en grande partie due à l'intégration d'outils pour QML.

Je vous laisse apprécier cette nouvelle version pleine de nouveautés dans votre langue maternelle.

=> Téléchargement de Qt Creator 2.1 Beta 2 : [Lien78](#)

Commentez cette news de Jonathan Courtois en ligne : [Lien79](#)

Nokia annonce ses objectifs de modularisation du framework Qt : serait-ce ce à quoi va ressembler Qt 4.8 ?

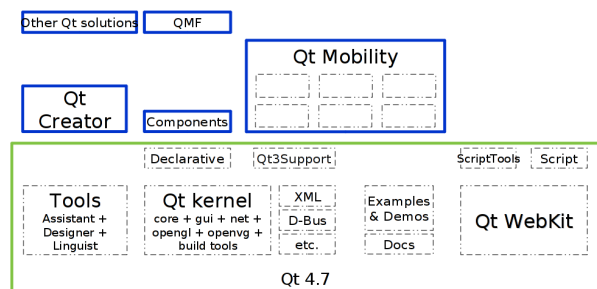
Un billet plus qu'intéressant est apparu dans les Qt Labs blogs de nos trolls favoris ([Lien80](#)). Il s'agit, ni plus ni moins, d'éclater le framework en une multitude de sous-projets.

Si, comme moi, vous suivez l'évolution de Qt depuis quelque temps, vous avez pu vous apercevoir du fait que des nouvelles technologies viennent le compléter version après version. Jusqu'ici, ces technologies étaient disponibles sous forme de module : le développeur pouvant choisir de n'inclure que ce dont il a besoin dans son application.

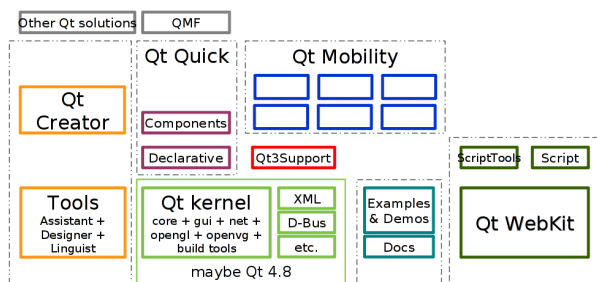
Nokia a décidé de pousser le concept un peu plus loin avec le projet « Qt Modularization » : éclater le dépôt, donner

plus d'indépendance aux mainteneurs respectifs et avoir des feuilles de route séparées. Ce n'est pas sans rappeler QtWebKit, qui a suivi cette voie depuis quelques mois déjà. Peut-être était-ce pour Nokia une façon de tester « grandeur nature » cette séparation.

Quoi qu'il en soit, rien de planifié de manière précise pour l'instant. Deux schémas sont fournis, mettant en opposition l'architecture actuelle et celle à laquelle on pourrait s'attendre.



Architecture Qt 4.7



Architecture possible - Qt 4.8 ?

De mon point de vue, modeste contributeur aux plugins SQL, cela signifierait un travail plus facile car je suis obligé de cloner l'intégralité du dépôt alors que moins de 1% me concerne. Je suis assez impatient de pouvoir tester un tel bijou.

Si le concept pouvait être poussé jusqu'à fournir des versions packagées de manière beaucoup plus atomique, alors là, je serais un geek heureux.

Commentez cette news d'Emmanuel Bourgerie en ligne : [Lien81](#)

Les derniers tutoriels et articles

État de HTML5 <canvas> dans QtWebKit

L'HTML5 introduit un nouvel élément de bloc nommé <canvas> pour vos pages Web. Cet article présente rapidement ses conformités et performances ainsi que de nombreuses démos. Il est important de noter que cet article concerne la branche trunk actuelle de QtWebKit, c'est pourquoi certaines parties ne sont pas pertinentes pour Qt 4.7.x.

Cet article est une traduction autorisée de State of HTML5 <canvas> in QtWebKit, par Andreas Kling ([Lien82](#)).

1. L'article original

Les *Qt Labs Blogs* sont des blogs tenus par les développeurs de Qt, concernant les nouveautés ou les utilisations un peu extrêmes du framework.

Nokia, Qt et leurs logos sont des marques déposées de *Nokia Corporation* en Finlande et/ou dans les autres pays. Les autres marques déposées sont détenues par leurs propriétaires respectifs.

Cet article est la traduction du billet State of HTML5 <canvas> in QtWebKit, par Andreas Kling ([Lien82](#)).

2. Introduction rapide

<canvas> est une nouvelle fonctionnalité dans HTML5 - un élément de bloc avec une API de dessin impératif. Il a été (un peu exagérément) présenté comme un « tueur de Flash ». Vous pouvez l'utiliser pour créer des pages Web graphiquement riches et des démonstrations de nouvelles technologies voient le jour quotidiennement. La fonctionnalité provient de WebKit, développée initialement pour le tableau de bord de Mac OS X. Naturellement, QtWebKit la supporte.



« Qt! » joliment écrit en utilisant la démo FlowerPower.

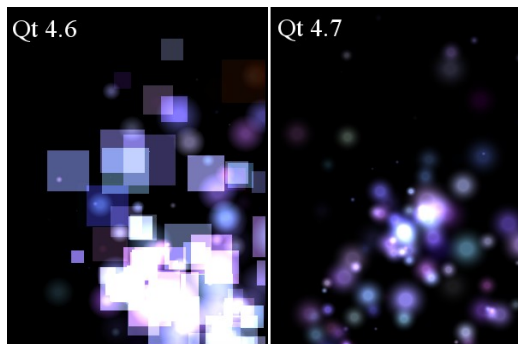
3. Conformité

La conformité des spécifications est élevée - nous passons plus de 90% de la suite de tests de Philip Taylor ([Lien83](#)) et il existe des correctifs à l'étude pour de nombreux échecs restants. En outre, certains des tests sont devenus invalides après des changements dans les spécifications. La plupart des erreurs de rendu connues ont été rectifiées et nous nous approchons d'une correspondance parfaite avec Chromium et Safari.

La grande exception est le cas des dégradés radiaux ; HTML5 définit ses dégradés radiaux sur la base de l'implémentation originale de CoreGraphics qui diffère des dégradés SVG standards (et c'est ce que QRadialGradient implémente). J'ai essayé de convaincre les développeurs

de WHATWG ([Lien84](#)) d'adopter les dégradés sur le style SVG pour les canevas, mais ils sentaient qu'il était trop tard pour un tel changement. Heureusement, je dois encore voir quelqu'un qui doit exploiter les possibilités supplémentaires des dégradés sur le style CoreGraphics « à l'état sauvage ».

Nous avons récemment obtenu le support pour les ombres floues grâce à Ariya Hidayat. Il s'agissait de la dernière grande fonctionnalité qui manquait dans notre implémentation.



L'une des nombreuses corrections de bogues depuis Qt 4.6, des captures d'écran de la démo Parcyle.

4. Performances

Les performances ont été en progression constante depuis la sortie de Qt 4.6 et dans Qt 4.7, vous verrez de grandes accélérations grâce aux efforts de Benjamin Poulain dans la vectorisation du code de mélange de QPainter. Nous ne sommes pas encore au niveau de Skia (boîte à outils de Chromium) dans le mélange de pixmapes lors d'agrandissements et/ou de transformations, mais Olivier Goffart travaille actuellement dessus et les chiffres préliminaires semblent bons.

Comme il n'existe pas d'objet « couleur » natif, les traits et les couleurs de remplissage sont gérés en utilisant des chaînes de caractères CSS. Auparavant, ils seraient passés par le parseur de CSS généré (lex) qui est très lent. J'ai récemment déposé un analyseur de couleurs rgba() optimisé qui aide à améliorer n'importe quelle application qui peint avec de nombreuses couleurs différentes.

La vitesse de manipulation des pixels est acceptable, pourtant nous n'avons pas encore atteint nos limites. Principalement parce que nous sommes obligés de faire la conversion RGB/BGR à la volée dans putImageData() et getImageData(). Ceci pourrait être atténué par l'ajout de QImage::Format_ABGR32_Premultiplied mais je pense

que ce serait exagéré pour le moment. De façon plus réaliste, on pourrait au moins vectoriser la conversion de format à l'intérieur de WebKit.

Notre plus grande faiblesse est le traceur de chemin - le traceur et le rastériser de chemin dans Qt sont lents par rapport aux autres boîtes à outils. Certaines personnes chez Google, travaillent sur des chemins accélérés par le GPU, ce que nous surveillons avec grand intérêt.

La seconde plus grande faiblesse est le rendu des dégradés. Qt met en cache les données de soixante dégradés au plus, mais il est très courant pour les applications à base de canevas d'utiliser beaucoup plus de dégradés uniques, ce qui rend le mécanisme de cache très coûteux. L'optimisation (et la vectorisation) de ces opérations est dans la file d'attente des projets de recherche.

5. Les travaux futurs

Les ingénieurs de Google travaillent sur une implémentation de canevas accélérée matériellement pour

WebKit. Espérant que nous serons en mesure de profiter de ce travail sur les plates-formes où ça sera significatif.

Actuellement, nous regardons l'optimisation de notre code de filtrage bilinéaire qui excelle actuellement dans de nombreux profils de performance.

6. Démon

Très bien, assez parlé. Voici quelques-unes de mes démon de <canvas> préférées pour le plaisir de vos yeux :

- Parcycle : [Lien85](#)
- Interferoplasma : [Lien86](#)
- Monster : [Lien87](#)
- Trail : [Lien88](#)
- Canvas Cycle : [Lien89](#)

Des tonnes de démon supplémentaires sur [canvasdemos.com](#) ([Lien90](#))

Retrouvez l'article d'Andreas Kling traduit par Jonathan Courtois en ligne : [Lien91](#)

Interview de Rich Green, CTO de Nokia

Qt Dev Days 2010 à Munich : le reportage

Question : est-il prévu de porter Qt sur des systèmes comme iPhone et Android, ou restera-t-il dédié aux périphériques Nokia ?

Rich : pour l'instant, nous allons nous concentrer sur Qt lui-même pour faire en sorte qu'il arrive avec de très bons outils pour les smartphones Nokia. Cela nous prend la majeure partie de notre temps et de nos ressources. Maintenant, si l'on regarde dans l'avenir, il y aura sûrement des opportunités de porter Qt sur d'autres plateformes que celles de Nokia et nous nous porterons très certainement sur les télévisions, les voitures, etc. qui pourront faire partie de l'écosystème de Nokia. En ce qui concerne iPhone et Android, nous n'avons pas de plan pour le moment les concernant, nous ne sommes pas contre, nous préférons tout simplement nous concentrer sur Symbian et MeeGo qui sont de très bonnes plateformes.

Question : en ce qui concerne Symbian, il était très difficile d'avoir une version récente sur un téléphone plus ancien, allez-vous faire quelque chose à propos de cela ?

Rich : la question de la compatibilité entre les versions est un problème difficile. Cependant, nous allons faire en sorte de fournir aux utilisateurs de Symbian le meilleur outil de mise à jour leur permettant d'avoir la dernière version du système d'exploitation, des applications, etc. Nous concentrons une bonne partie de nos efforts sur la refonte de ce système de mise à jour, ce n'est pas un changement facile mais nous pensons qu'il peut apporter une grande valeur ajoutée à nos utilisateurs. Il y a bien sûr un point où la technologie est trop vieille pour être capable de faire fonctionner une nouvelle plateforme, cependant nous pouvons faire mieux que ce qui existe actuellement.

Question : il a été annoncé que les efforts pour Qt 4.7 étaient portés sur les performances et l'amélioration de l'existant. Cependant nous voyons de nombreuses

nouvelles technologies telles que QML, Scene Graphe, Qt/3D, etc. quel est le véritable objectif des nouvelles versions de Qt ?

Rich : la rapidité de développement d'une application avec Qt Quick est un élément important de l'accélération, non pas de l'application, mais du temps de développement et nous devons fournir quelque chose de tel à nos développeurs. D'un point de vue visuel, les applications devaient également être aussi attirantes que ce qui existe ailleurs, c'est pourquoi nous avons mis nos efforts sur les deux, à savoir Qt Quick et les performances de l'existant.

Question : qu'est-ce que vous pensez du futur de Symbian et MeeGo, vont-ils continuer séparément ou MeeGo risque-t-il de prendre le pas sur Symbian dans un futur éloigné ?

Rich : c'est trop tôt pour s'avancer sur cette question, nous investissons sur les deux plateformes et nous avons un réel succès avec Symbian^3 sur le N8, donc pour l'instant nous allons garder cela dans ce sens. En même temps, nous faisons partie du programme MeeGo avec Intel et il s'agit tous les deux (avec Symbian) de programmes importants pour Nokia et je ne pense pas que nous ayons à choisir. Notre objectif est de faire fonctionner aussi bien Qt et QML sur ces deux systèmes d'exploitation pour que les développeurs puissent choisir en fonction du marché.

Question : quand MeeGo sera-t-il livré pour la première fois sur un smartphone Nokia ?

Rich : nous n'allons pas faire de spéculations, nous parlerons de la sortie de MeeGo lorsqu'il sera prêt.

Retrouvez l'interview de Rich Green sur le reportage des Qt Dev Days 2010 à Munich en ligne : [Lien92](#)

Windows Phone

Les derniers tutoriels et articles



Tutoriel : introduction au développement d'applications Windows Phone 7

Cet article constitue une introduction au développement d'applications pour Windows Phone 7. Il est le premier d'une série traitant de divers aspects de programmation sous ce nouvel environnement.

1. Introduction

Windows Phone 7 (WP7) est la nouvelle plateforme de développement de Microsoft destinée aux smartphones.

Dans ce premier tutoriel, vous découvrirez comment démarrer le développement d'applications avec Windows Phone 7.

Nous commencerons par installer les outils nécessaires au développement sur WP7 puis nous créerons un premier programme simpliste en utilisant Silverlight pour WP7.

Enfin, nous l'exécuterons sur l'émulateur et découvrirons son fonctionnement.

2. Généralités sur WP7

WP7 supporte deux langages pour développer des applications :

- C# ;
- VB.NET.

WP7 supporte également deux plateformes de développement :

- Silverlight ;
- XNA.

Silverlight, qui a déjà fait ses preuves pour le développement d'applications client léger, est en général utilisé dans le développement d'applications de gestion. La puissance du XAML, des mécanismes de bindings ainsi que la richesse des contrôles visuels (textbox, button...) en font un langage de premier choix pour le développement d'applications utilitaires.

XNA quant à lui est la plateforme de développement de jeux. On peut aussi bien développer des jeux 2D que 3D. XNA est aussi plus performant que Silverlight quand il s'agit de faire du traitement audio. Bref, il est particulièrement adapté au développement de jeux sur smartphones.

Dans les premiers tutoriels nous nous concentrerons sur l'utilisation de Silverlight pour WP7 mais dans les prochains tutoriels nous découvrirons également le développement XNA.

Pour le reste de ces articles, je considère que vous avez déjà des notions dans le développement d'applications utilisant le XAML et le C# sinon n'hésitez pas à consulter les tutoriels de la rubrique concernée : [Lien93](#).

Notez que Silverlight pour WP7 n'est pas tout à fait le Silverlight que vous avez déjà pu rencontrer. C'est un

Silverlight adapté aux Windows Phones. Il s'agit d'une autre version de Silverlight, basée à ce jour sur Silverlight 3 et dont les contrôles standards ont été adaptés pour prendre en charge les spécificités des téléphones. Ce qui implique que si un contrôle n'a pas été écrit dans ce sens, alors il ne fonctionnera pas avec WP7. On ne pourra donc pas récupérer la pléthore de contrôles déjà existants.

Une dernière remarque par rapport au matériel. Au jour d'aujourd'hui nous avons des téléphones qui peuvent supporter les résolutions suivantes :

- - Large : 480 x 800 ;
- - Small : 320 x 480.

et ils fonctionnent aussi bien en mode portrait qu'en mode paysage.

3. Installation des outils

On commence par télécharger les Windows Phone Developer Tools : [Lien94](#).

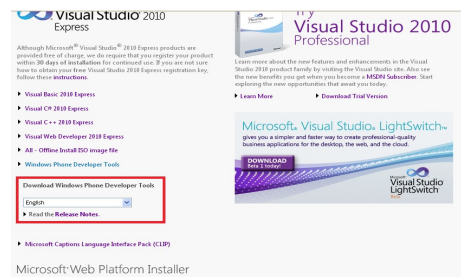


Figure 1 : télécharger les Windows Phone Developer Tools.

À noter qu'il faut disposer d'une version de Windows Vista ou bien de Windows 7, sinon vous aurez l'erreur suivante :



Figure 2 : pré-requis : Windows Vista ou Windows 7.

Une fois ces conditions réunies, l'installation peut débuter :



Figure 3 : installation des Windows Phone Developer Tools.

À noter qu'il faut être connecté à Internet car le programme télécharge des composants :

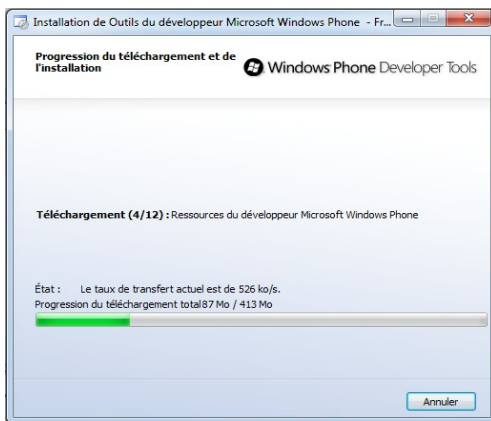


Figure 4 : téléchargement des composants de l'installation.

On enchaîne ensuite sur l'installation de ces composants :

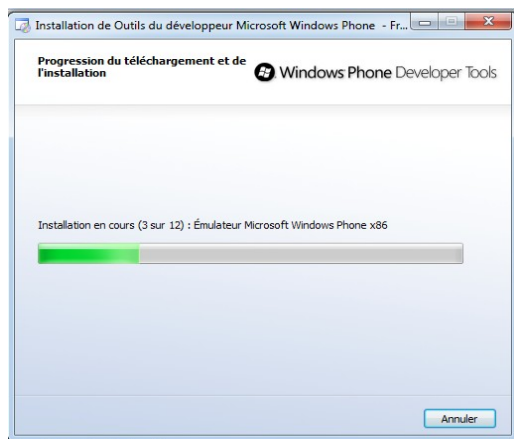


Figure 5 : installation des composants.

À la fin de l'installation, on pourra découvrir un nouvel élément dans le menu démarrer :

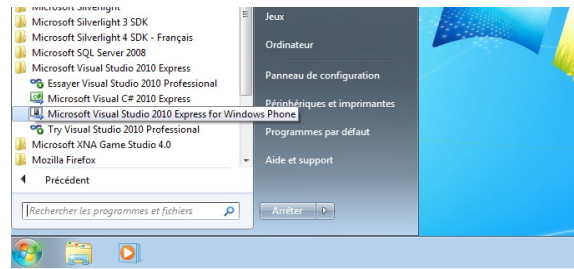


Figure 6 : Visual Studio for Windows Phone.

Et démarrons notre nouveau Visual Studio for Windows Phone...

4. Première application Windows Phone 7

Passons maintenant à la création de notre premier projet Windows Phone 7.

Cliquons sur Fichier --> Nouveau Projet et choisissons une "Application Windows Phone" :

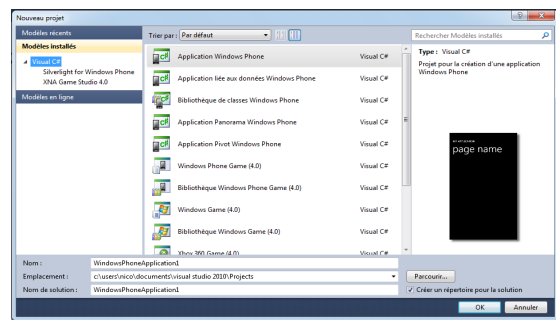


Figure 7 : création d'une application Windows Phone.

Visual Studio génère plein de choses et on observe la fenêtre de design qui contient à droite le XAML généré et à gauche une prévisualisation du XAML "embarqué" dans une image de téléphone.

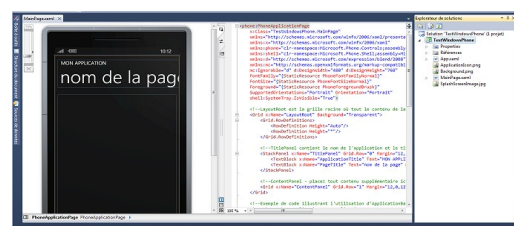


Figure 8 : le designer de Visual Studio for Windows Phone.

Avant de regarder ce qui a été généré, on vérifie que tout marche bien. Compilons puis démarrons l'application (F5).

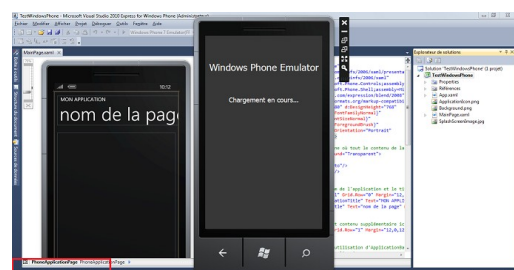


Figure 9 : démarrage de l'émulateur Windows Phone 7.

L'émulateur se lance et on peut voir dans le cadre rouge que Visual Studio est en train de se connecter à l'émulateur. L'application n'est pas encore chargée. Il faudra attendre que Visual Studio se mette en debug et que l'on voie marqué en bas à droite « déploiement réussi » pour que l'application soit effectivement chargée :

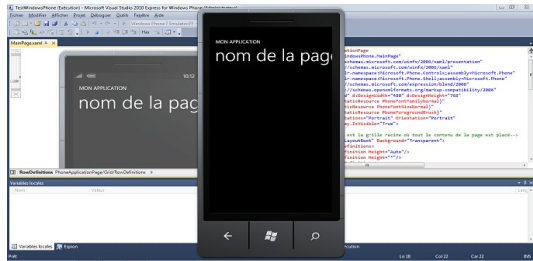


Figure 10 : l'application est prête à être utilisée dans l'émulateur.

5. L'émulateur

Jouons un peu avec l'émulateur. Nous avons des icônes à droite de l'émulateur qui apparaissent :

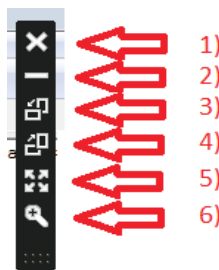


Figure 11 : icônes de contrôle de l'émulateur.

- 1) permet de fermer l'émulateur ;
- 2) permet de réduire l'émulateur ;
- 3) permet de faire pivoter l'émulateur pour passer en mode paysage ;

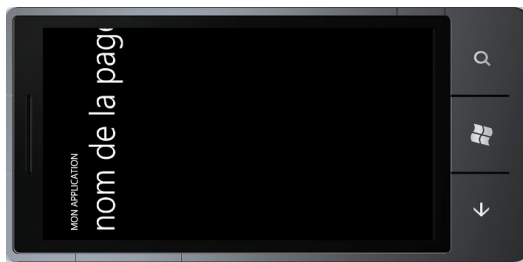


Figure 12 : émulateur en mode paysage.

- 4) comme le 3) mais dans l'autre sens ;
- 5) permet d'ajuster l'émulateur à la taille de l'écran ;
- 6) permet de définir le zoom de l'émulateur.

On peut également constater des boutons en bas de l'émulateur, avec à gauche le bouton "back", au milieu le bouton "start" et un bouton "rechercher".

Le bouton "back" correspond à la fermeture de l'application. On peut le constater lorsque l'application est en mode débogage ; si l'on clique sur "back" alors le débogueur de Visual Studio s'arrête.

Cependant, l'émulateur est toujours lancé et chargé en mémoire. Visual Studio pourra s'y reconnecter lors du prochain démarrage de l'application (F5). Ceci s'avère bien utile et optimisera les prochains lancements de

l'application dans l'émulateur. En effet, il ne sera plus utile à l'émulateur de faire sa phase de synchronisation...

Le bouton de l'émulateur qui servira sans doute le plus est le bouton qui permet de passer d'un mode paysage à un mode portrait et inversement. Nous pourrions ainsi constater comment l'application se comporte dans les deux modes.

En l'occurrence, on peut voir ici que ce n'est pas joli joli... En effet, ce comportement n'est pas géré par l'application générée par défaut.

6. Analyse du code

Repassons sur Visual Studio et regardons ce qui est généré.

En ouvrant le fichier App.xaml.cs, on se rend compte que la classe contient une propriété :

```
App.xaml.cs
public PhoneApplicationFrame RootFrame
{ get; private set; }
```

La propriété RootFrame identifie la page de démarrage de l'application. C'est le container de base pour toutes les applications Windows Phone. Elle est de type PhoneApplicationFrame ([Lien95](#)) et permet d'accueillir des pages pour Windows Phone, de type PhoneApplicationPage ([Lien96](#)).

On voit aussi apparaître un code qu'on ne voit pas dans les applications Silverlight usuelles :

```
App.xaml.cs
// Ne pas ajouter de code supplémentaire à cette méthode
private void InitializePhoneApplication()
{
    if (phoneApplicationInitialized)
        return;

    // Créez le frame, mais ne le définissez pas encore comme RootVisual ; cela permet à l'écran de
    // démarrage de rester actif jusqu'à ce que l'application soit prête pour le rendu.
    RootFrame = new PhoneApplicationFrame();
    RootFrame.Navigated += CompleteInitializePhoneApplication;

    // Gérer les erreurs de navigation
    RootFrame.NavigationFailed += RootFrame_NavigationFailed;

    // Garantir de ne pas retenter l'initialisation
    phoneApplicationInitialized = true;
}

// Ne pas ajouter de code supplémentaire à cette méthode
private void CompleteInitializePhoneApplication(object sender, NavigationEventArgs e)
{
    // Définir le Visual racine pour permettre à l'application d'effectuer le rendu
    if (RootVisual != RootFrame)
        RootVisual = RootFrame;
}
```

```
// Supprimer ce gestionnaire, puisqu'il est
devenu inutile
RootFrame.Navigated -=
CompleteInitializePhoneApplication;
}
```

Il s'agit de tout ce qui est initialisation propre au téléphone.

Allons voir le fichier MainPage.xaml.cs, nous pouvons voir que la page hérite de PhoneApplicationPage :

```
MainPage.xaml.cs
public partial class MainPage :
PhoneApplicationPage
{
}
```

Quand on ouvre le fichier MainPage.xaml, un point important à remarquer est que ça ressemble beaucoup à une application Silverlight classique. On voit une Grid, un TextBlock, etc.

Modifions-les un peu pour afficher quelque chose d'un peu plus sympa que "mon application" ou "nom de la page" en changeant les deux textblocks :

```
MainPage.xaml
<TextBlock x:Name="ApplicationTitle" Text="Test
de Windows Phone 7" Style="{StaticResource
PhoneTextNormalStyle}" />
<TextBlock x:Name="PageTitle" Text="Hello
World !" Margin="9,-7,0,0" Style="{StaticResource
PhoneTextTitle1Style}" />
```

En haut du fichier, nous remarquons aussi la déclaration suivante :

```
MainPage.xaml
SupportedOrientations="Portrait"
Orientation="Portrait"
```

qui signifie, comme on peut le deviner, que l'application démarre en mode portrait et ne supporte que celui-ci.

Un petit coup d'intellisense et on remarque que les modes supportés sont :

- Portrait ;
- Landscape ;
- PortraitOrLandscape.

Si je passe en mode paysage (Landscape), j'aurai bien sûr l'affichage suivant :



Figure 13 : l'application en mode paysage.

Réglons enfin sur :

```
MainPage.xaml
SupportedOrientations="PortraitOrLandscape"
Orientation="Portrait"
```

et rajoutons un textblock :

```
MainPage.xaml
<TextBlock x:Name="OrientationTextBlock"
FontSize="25" />
```

Il est possible de détecter un changement d'orientation en surchargeant la méthode OnOrientationChanged :

```
MainPage.xaml.cs
public partial class MainPage :
PhoneApplicationPage
{
// Constructeur
public MainPage ()
{
InitializeComponent();
}

protected override void
OnOrientationChanged(OrientationChangedEventArgs
e)
{
OrientationTextBlock.Text = "On passe en mode
" + e.Orientation ;
base.OnOrientationChanged(e);
}
}
```

À chaque changement d'orientation, on pourra voir un message signalant la réception de l'événement et la réorientation des contrôles :



Figure 14 : l'application change de mode d'orientation.

Bien sûr, c'est l'endroit idéal pour recalculer les dimensions si l'on souhaite que son application ait un look qui va bien dans le mode choisi...

On peut voir enfin dans la solution, des images comme ApplicationIcon.png. Si vous les ouvrez vous pourrez notamment constater que ce sont ces images qu'il faut changer si l'on souhaite changer, par exemple, l'image de l'application :

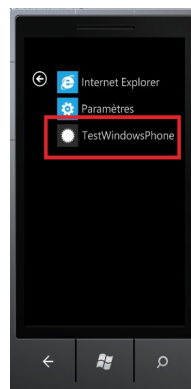


Figure 15 : l'image de l'application.

7. Les autres templates de création de projet

Notez que lors de la création du projet vous avez pu voir différents templates pour créer une application pour WP7. J'ai choisi le plus basique : **Application Windows Phone** car c'est le plus simple et que je trouve qu'il est plus intéressant pour apprendre et rajouter des briques petit à petit à son application en comprenant vraiment ce que l'on fait.

Vous pouvez voir ci-dessous la liste complète des templates proposés :

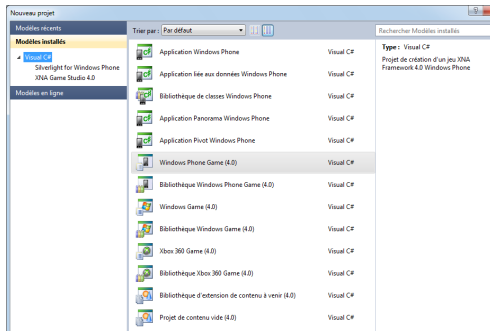


Figure 16 : la liste des templates de création de projet.

Le template **Application liée aux données Windows Phone** génère un projet exemple qui contient une ListBox bindée à des données. On voit également comment naviguer entre des pages de l'application lors d'un clic sur un élément de la ListBox.

Le template **Bibliothèque de classes Windows Phone** est un projet que l'on utilisera pour rajouter des composants à son application (nouvelle bibliothèque de contrôles, nouveaux modules, classes, etc.).

Le template **Application Panorama** présente un des contrôles importants dans les applications WP7 qui permet de faciliter la consultation de l'affichage d'un écran trop grand. Il permet de scinder l'écran et propose une navigation comme si l'on tournait les pages d'un livre. Un glissement horizontal sur l'écran (swipe) permet de passer à l'élément suivant. Lorsque l'on atteint le dernier élément on repasse automatiquement au premier.

Le template **Application Pivot Windows Phone** présente

l'autre contrôle important dans le développement d'applications WP7. Il permet grosso modo la même chose que le contrôle Panorama. La différence est qu'on utilisera plutôt le contrôle **Pivot** lorsqu'on veut présenter plusieurs pages de la même donnée.

Le contrôle **Panorama** sera plutôt utilisé lorsqu'on veut présenter une navigation entre plusieurs pages.

Le template **Windows Phone Game** permet de développer un jeu avec XNA. Ici le paradigme est différent d'une application Silverlight et les références se font au framework XNA. Le projet généré ne fait qu'afficher simplement un fond bleu mais présente la structure basique d'un jeu utilisant XNA.

Le template **Bibliothèque Windows Phone Game** permet de rajouter des composants à son application XNA. Notez que nous reviendrons sur la programmation XNA dans un tutoriel ultérieur.

Les autres templates sont des templates XNA pour des développements autres que téléphone (XBox, etc.).

N'hésitez pas à fouiller dans les projets générés par les autres templates, ils ont des vertus didactiques et sont toujours intéressants à comprendre.

8. Téléchargements

Vous pouvez télécharger ici les sources du projet : version rar (63 Ko) ([Lien97](#)), version zip (69 Ko) ([Lien98](#)).

9. Conclusion

Ce premier tutoriel a donc constitué une introduction au développement d'applications pour Windows Phone. D'autres suivront. Vous avez pu découvrir comment installer les différents outils pour développer sur WP7.

Vous avez également pu voir comment créer une application simple et voir son exécution dans l'émulateur Windows Phone.

Enfin, nous avons également eu un aperçu des différentes orientations du téléphone et le moyen de les détecter.

J'espère que ce tutoriel a pu vous être utile et vous a donné envie de créer des applications sur Windows Phone 7.

Retrouvez l'article de Nico-pyright(c) en ligne : [Lien99](#)

Tutoriel : Utiliser la ListBox et l'Isolated Storage dans vos applications Windows Phone 7 avec Silverlight

Cet article est la suite du tutoriel d'introduction au développement d'applications Windows Phone 7 (ci-dessus). Il présentera d'autres aspects du développement WP7 avec Silverlight en explorant notamment la ListBox et l'Isolated Storage.

1. Introduction

Windows Phone 7 (WP7) est la nouvelle plateforme de développement de Microsoft destinée aux smartphones.

Si vous découvrez la programmation WP7, n'hésitez pas à consulter l'article précédent : tutoriel d'introduction au développement d'applications Windows Phone 7 (ci-dessus) ([Lien99](#)).

Dans ce deuxième tutoriel, vous découvrirez la puissance du contrôle ListBox WP7 et comment il s'adapte au développement sur téléphone.

Vous verrez aussi comment traiter des événements simples, comme un clic sur un bouton par exemple.

Vous verrez enfin comment faire persister de l'information entre les différents démarrages du téléphone et lorsque l'application est suspendue.

2. Utiliser la ListBox WP7

Je vous ai indiqué dans le tutoriel précédent que les contrôles standards ont été réécrits pour tirer parti des fonctionnalités du téléphone.

C'est le cas de la ListBox qui prend en charge automatiquement un effet classique dans les téléphones : le défilement vertical.

Vous allez voir que le rendu est plutôt élégant et qu'il n'y a rien à faire...

Commençons par créer une nouvelle application comme on l'a déjà fait dans le tutoriel précédent : **Fichier -> Nouveau projet -> Application Windows Phone**. J'appelle cette application : ListBoxDemo.

Nous allons développer une application toute simple de "todo list".

Dans le Xaml, il s'agit de rajouter un TextBox pour saisir une valeur et un bouton pour ajouter la valeur saisie.

Ensuite, nous pouvons ajouter une ListBox qui contiendra un TextBlock pour afficher les valeurs de la liste de TODO :

MainPage.xaml

```
<Grid x:Name="LayoutRoot"
Background="Transparent">
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="*" />
  </Grid.RowDefinitions>

  <StackPanel x:Name="TitlePanel" Grid.Row="0"
Margin="12,17,0,28">
    <TextBlock x:Name="ApplicationTitle"
Text="Demo ListBox" Style="{StaticResource
PhoneTextNormalStyle}" />
  </StackPanel>

  <Grid x:Name="ContentPanel" Grid.Row="1"
Margin="12,0,12,0">
    <!--<TextBlock Text="Nom : " Margin="0 20 43
0" FontSize="35" />-->
    <Grid>
      <TextBlock Text="TODO : " Margin="0 20"
FontSize="45" />
      <TextBox Margin="150 15 0 0" Height="80"
VerticalAlignment="Top" FontSize="30"
Name="todoTb" Width="320"
HorizontalAlignment="Left" />
      <Button Height="70" Width="250"
VerticalAlignment="Top" Margin="0 100 0 0"
Content="Ajouter" />
      <ListBox Margin="0 190 0 0" Name="listbox">
        <ListBox.ItemTemplate>
          <DataTemplate>
            <Border Background="#FFDEDE"
CornerRadius="10" Margin="20">
              <StackPanel Width="400">
                <TextBlock
HorizontalAlignment="Center" Text="{Binding
Todo}"
FontSize="45"
Foreground="Blue" Margin="20" />
              </StackPanel>
            </Border>
          </DataTemplate>
        </ListBox.ItemTemplate>
      </ListBox>
    </Grid>
  </Grid>
```

```
</ListBox>
</Grid>
</Grid>
</Grid>
```

Si vous démarrez l'application, vous pourrez noter que la TextBox prend automatiquement en charge l'affichage d'un clavier virtuel et qu'il n'y a rien à faire de particulier.



Figure 1 : Le clavier virtuel.

En regardant ce Xaml, on observe notamment le TextBox "todoTb" ainsi qu'un bouton qui possède une propriété Click : **Click="Button_Click"**.

On peut ainsi constater que l'appui sur un bouton (le "tap") peut être géré par le même handler que pour une application Silverlight pilotée à la souris.

Enfin, nous pouvons voir la ListBox "listbox" qui possède un TextBlock dans sa propriété ItemTemplate avec une valeur liée à la propriété Todo : **Text="{Binding Todo}"**.

Pour assurer le binding, créons une classe TodoElement qui implémente [INotifyPropertyChanged](#) ([Lien100](#)) :

TodoElement.cs

```
public class TodoElement : INotifyPropertyChanged
{
  private string todoElement;

  public event PropertyChangedEventHandler
PropertyChanged;

  public void NotitfyPropertyChanged(string
propertyName)
  {
    if (PropertyChanged != null)
    {
      PropertyChanged(this, new
PropertyChangedEventArgs (propertyName));
    }
  }

  public string Todo
  {
    get { return todoElement; }
    set
    {
      todoElement = value;
      NotitfyPropertyChanged ("Todo");
    }
  }
}
```

Et assurons ce binding dans le code behind via la propriété ItemsSource de la ListBox grâce à une ObservableCollection de TodoElement :

```

MainPage.xaml.cs
public partial class MainPage :
PhoneApplicationPage
{
    ObservableCollection<TodoElement> listTodo =
new ObservableCollection<TodoElement>();

    public MainPage()
    {
        InitializeComponent();

        listbox.ItemsSource = listTodo;
    }

    private void Button_Click(object sender,
RoutedEventArgs e)
    {
        listTodo.Add(new TodoElement { Todo =
todoTb.Text });
    }
}

```

3. Gestion de l'événement d'appui sur le bouton

Nous voyons également la méthode Button_Click qui permettra de gérer l'événement d'appui sur le bouton. Ici, nous rajouterons un élément à la liste.

On peut voir que la gestion de l'événement d'appui sur une touche se passe de la même façon que la gestion du clic sur une application Silverlight classique. On verra lors de prochains tutoriels qu'il est possible de gérer les événements utilisateurs d'une autre façon, pour reconnaître par exemple les "glisser" (swipe) ou les gestes avec une structure plus complexe.

Pour des applications simples de gestion, comme ici, on aura tout à fait intérêt à utiliser cette méthode, en utilisant les handler de clic sur un bouton par exemple.

4. Démarrage de l'application dans l'émulateur

Démarrons l'application dans l'émulateur (F5) et ajoutons quelques éléments...
Ce qui donne :



Figure 2 : Listbox avant défilement vertical.

Et si on fait défiler vers le bas en maintenant la pression sur la souris et en faisant glisser vers le haut, on voit le

défilement attendu ; que vous pouvez retrouver sur la vidéo ci-dessous (voir la vidéo en ligne : [Lien101](#)).

Vous conviendrez que l'effet est plutôt réussi. Et tout ça sans code supplémentaire. La Listbox WP7 intègre par défaut ce comportement, ce qui est bien pratique.

Cliquons sur "back", qui a pour effet de terminer l'application, et relançons-la.

Horreur ! Tout a disparu. Ceci est plutôt dérangeant pour une liste de TODO. Si on doit se rappeler des tâches à faire et les ressaisir à chaque fois, l'application perd grandement de son intérêt...

Forcément me diriez-vous, nous n'avons rien sauvegardé. Voyons comment faire.

5. Persistance dans l'Isolated Storage

Pour faire persister de l'information, nous avons à notre disposition l'Isolated Storage. Le principe sera de sérialiser notre liste de TODO dans l'Isolated Storage au moment où l'on souhaite la sauvegarder. De même, nous pourrions la désérialiser quand l'on souhaitera recharger les informations.

Pour ce faire, on va utiliser une classe helper qui est fournie dans le training kit de WP7 ([Lien102](#)) qui permet de sérialiser et désérialiser en JSON.

Nous aurons besoin pour ce faire de rajouter une référence à System.ServiceModel.Web.

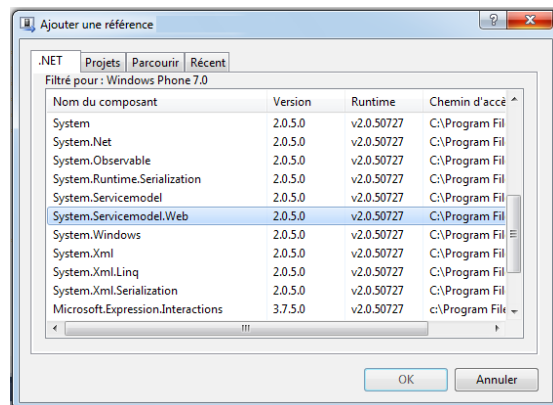


Figure 3 : Ajout de la référence à System.ServiceModel.Web.dll.

On ajoute la classe IsolatedStorageHelper dont le code ci-dessous est assez simple à comprendre :

```

IsolatedStorageHelper.cs
// -----
// Microsoft Developer & Platform Evangelism
//
// Copyright (c) Microsoft Corporation. All rights reserved.
//
// THIS CODE AND INFORMATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
// EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES
// OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR PURPOSE.
// -----
// The example companies, organizations, products, domain names,

```

```

// e-mail addresses, logos, people, places, and
events depicted
// herein are fictitious. No association with
any real company,
// organization, product, domain name, email
address, logo, person,
// places, or events is intended or should be
inferred.
// -----

using System.IO;
using System.IO.IsolatedStorage;
using System.Runtime.Serialization.Json;
using System.Text;

namespace ListBoxDemo
{
    public static class IsolatedStorageHelper
    {
        public static T GetObject<T>(string key)
        {
            if
(IsolatedStorageSettings.ApplicationSettings.Cont
ains(key))
            {
                string serializedObject =
IsolatedStorageSettings.ApplicationSettings[key].
ToString();
                return Deserialize<T>(serializedObject);
            }

            return default(T);
        }

        public static void SaveObject<T>(string key,
T objectToSave)
        {
            string serializedObject =
Serialize(objectToSave);
            IsolatedStorageSettings.ApplicationSettings
[key] = serializedObject;
        }

        public static void DeleteObject(string key)
        {
            IsolatedStorageSettings.ApplicationSettings
.Remove(key);
        }

        private static string Serialize(object
objectToSerialize)
        {
            using (MemoryStream ms = new
MemoryStream())
            {
                DataContractJsonSerializer serializer =
new
DataContractJsonSerializer(objectToSerialize.GetT
ype());
                serializer.WriteObject(ms,
objectToSerialize);
                ms.Position = 0;

                using (StreamReader reader = new
StreamReader(ms))
                {
                    return reader.ReadToEnd();
                }
            }
        }

        private static T Deserialize<T>(string

```

```

jsonString)
    {
        using (MemoryStream ms = new
MemoryStream(Encoding.Unicode.GetBytes(jsonString
)))
        {
            DataContractJsonSerializer serializer =
new DataContractJsonSerializer(typeof(T));
            return (T)serializer.ReadObject(ms);
        }
    }
}

```

J'ai choisi de sauvegarder les informations après chaque ajout à la liste de TODO, mais il pourrait tout à fait être envisageable de faire les sauvegardes lors de l'événement de fermeture de l'application.

Ce qui nous donne :

MainPage.xaml.cs

```

private void Button_Click(object sender,
RoutedEventArgs e)
{
    listTodo.Add(new TodoElement { Todo =
todoTb.Text });

    IsolatedStorageHelper.SaveObject("ListeTodo",
listTodo);
}

```

Pour recharger les informations, on peut modifier le constructeur de MainPage pour avoir :

MainPage.xaml.cs

```

List<TodoElement> todoElementList;
ObservableCollection<TodoElement> listTodo = new
ObservableCollection<TodoElement>();

public MainPage()
{
    InitializeComponent();

    todoElementList =
IsolatedStorageHelper.GetObject<List<TodoElement>
>("ListeTodo");
    if (todoElementList == null)
        todoElementList = new List<TodoElement>();

    foreach (TodoElement todoElement in
todoElementList)
    {
        listTodo.Add(todoElement);
    }
    listbox.ItemsSource = listTodo;
}

```

Redémarrons l'application et saisissons quelques valeurs.

Cliquons sur back, le débogueur se ferme. Relançons l'application ainsi que le débogueur (F5) et ô miracle, notre Listbox est remplie avec les informations déjà saisies.

Merci la sérialisation.

Il ne manquera plus à notre application qu'à savoir supprimer des éléments de la liste pour qu'elle soit vraiment fonctionnelle.

Ceci est plutôt simple à faire, il faudrait rajouter un bouton dans l'ItemTemplate de la Listbox, brancher un événement click dessus et supprimer de la liste le todo sélectionné. Comme ce n'est pas l'objet du tutoriel, nous n'allons pas le faire. Je vous encourage chaudement à essayer afin de vous entraîner.

Juste pour le plaisir, n'hésitez pas à passer en debug dans l'IsolatedStorageHelper, vous pourrez voir notamment comment est sérialisée notre liste de Todo. Par exemple :

MainPage.xaml.cs

```
[{"Todo": "aaa"}, {"Todo": "bbb"}]
```

est la représentation JSON d'une liste contenant deux todo : "aaa" et "bbb".

6. Réactivation d'une application

Disons que notre application est fonctionnelle et que je suis en train de noter un élément à faire et que je reçois un appel. Ou alors, j'ai besoin d'aller chercher une information sur Internet pour compléter ma saisie. Ce qui peut se résumer par le scénario suivant :

Je lance mon application. Je commence à saisir. J'appuie sur start pour aller dans le navigateur, je fais ma recherche, je ferme la recherche et je reviens sur mon application. Malheur, ce que j'étais en train de saisir a disparu.

C'est le principe de ce qu'on appelle le Tombstone, qui correspond au passage en mode "désactivé" de l'application.

Ce qui se passe en fait c'est que lorsque l'application passe en mode désactivé puis repasse en mode réactivé, alors l'application est relancée depuis le début (ce qui explique que l'on perde nos données saisies) mais à la différence près que l'on passe dans un événement particulier, signe de la réactivation de l'application.

Le modèle de développement de Windows Phone 7 nous offre l'opportunité de stocker des informations temporaires pendant le temps où l'application est désactivée et où elle se réactive.

Pour bien comprendre comment cela fonctionne, ouvrons le fichier App.Xaml.cs et mettons un point d'arrêt dans les méthodes suivantes :

App.xaml.cs

```
// Code à exécuter lorsque l'application démarre (par exemple, à partir de Démarrer)
// Ce code ne s'exécute pas lorsque l'application est réactivée
private void Application_Launching(object sender, LaunchingEventArgs e)
{
}

// Code à exécuter lorsque l'application est activée (affichée au premier plan)
// Ce code ne s'exécute pas lorsque l'application est démarrée pour la première fois
private void Application_Activated(object sender, ActivatedEventArgs e)
{
}
```

```
// Code à exécuter lorsque l'application est désactivée (envoyée à l'arrière-plan)
// Ce code ne s'exécute pas lors de la fermeture de l'application
private void Application_Deactivated(object sender, DeactivatedEventArgs e)
{
}

// Code à exécuter lors de la fermeture de l'application (par exemple, lorsque l'utilisateur clique sur Précédent)
// Ce code ne s'exécute pas lorsque l'application est désactivée
private void Application_Closing(object sender, ClosingEventArgs e)
{
}
```

Relancez l'application en debug. On passe dans un premier temps dans l'événement Application_Launching. Cliquons sur Start et l'événement Application_Deactivated est levé. Cliquons sur back et nous pouvons observer l'affichage d'un texte "reprise" dans l'émulateur :

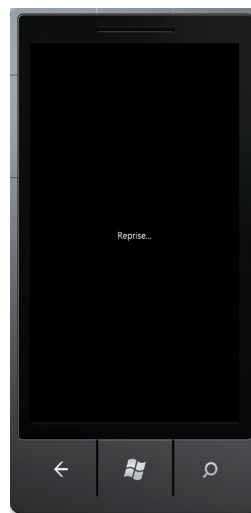


Figure 4 : Reprise de l'application avec désactivation.

Puis nous passons dans l'événement Application_Activated.

Cliquons sur back et nous passons dans l'événement Application_Closing.

Pour faire persister temporairement la valeur de la textbox, nous pourrions utiliser le dictionnaire State ([Lien103](#)) du PhoneApplicationService.

Pour ce faire, rajoutons un événement de changement de valeur pour la textbox :

MainPage.xaml

```
<TextBox Name="todoTb" [...]
TextChanged="todoTb_TextChanged" />
```

et dans le code behind :

MainPage.xaml.cs

```
private void todoTb_TextChanged(object sender, TextChangedEventArgs e)
{
    PhoneApplicationService.Current.State["CurrentTodo"] = todoTb.Text;
}
```

Cela nous permet de stocker les informations dans le dictionnaire d'états temporaires.

Pour récupérer les informations, on peut le faire dans le constructeur de la page par exemple :

```

MainPage.xaml.cs
public MainPage()
{
    InitializeComponent();

    if (PhoneApplicationService.Current.State.
        ContainsKey("CurrentTodo"))
        todoTb.Text = (string)
        PhoneApplicationService.Current.State
        ["CurrentTodo"];

    ...
}

```

Recommençons l'opération.

On lance l'application. On clique sur Start, on clique sur Back et on peut constater que la valeur de la TextBox est bien restaurée.

Parfait et cela évitera à nos utilisateurs d'être frustrés si jamais ils ont saisi un long texte...

Notez que nous avons utilisé la persistance au niveau de l'application. Il est également possible d'utiliser la persistance au niveau de la page ; en effet, la PhoneApplicationPage dispose aussi d'un dictionnaire State ([Lien104](#)). Celui-ci sera plutôt utilisé via les

événements de navigation (OnNavigatedFrom ([Lien105](#)) et OnNavigatedTo ([Lien106](#))) pour faire persister des informations d'affichage (focus, etc.) lors de la navigation.

7. Remarques

La ListBox est un contrôle très puissant, mais elle peut souffrir de lenteur, surtout si elle est très volumineuse.

On pourra avantageusement tirer parti du contrôle écrit par David Anson, le DeferredLoadListBox qui pallie ces problèmes de lenteur en se basant sur des StackPanel ([Lien107](#)).

8. Téléchargements

Vous pouvez télécharger ici les sources du projet : version rar (75 Ko) ([Lien108](#)), version zip (81 Ko) ([Lien109](#)).

9. Conclusion

Ce tutoriel a donc présenté comment la ListBox WP7 fonctionnait sur les téléphones. Nous avons également vu comment réagir à un appui sur un bouton. Enfin, nous avons eu un aperçu de comment faire persister des informations entre les lancements d'applications et lorsque celles-ci passent en mode désactivé.

J'espère que ce tutoriel a pu vous être utile et vous a donné envie de vous lancer dans la création d'applications sur Windows Phone 7.

Retrouvez l'article de Nico-pyright(c) en ligne : [Lien101](#)



Objective-C pour le développeur avancé - Le langage iPhone/iPad et Mac OS X pour les développeurs C++/Java/C#

Face à un C++ puissant, efficace et maîtrisé, Objective-C surprend par sa richesse et sa souplesse.

Adressé au développeur confirmé, ce livre dense et érudit guidera les amoureux de la programmation iPhone/iPad et Mac OS X à travers toutes les subtilités de ce langage.

Objective-C, langage objet indispensable pour développer en natif sous Mac OS X et pour l'iPhone et l'iPad

Avec le succès de l'iPhone et de l'iPad, la maîtrise d'Objective-C, langage natif des systèmes Apple Mac OS X et iPhone/iPad, devient un passage obligé pour les professionnels de la programmation - alors même qu'il ne fait pas partie de la formation classique des développeurs.

Adressé au développeur qui connaît déjà d'autres langages objet, cet ouvrage éclaire toutes les subtilités d'Objective-C en le comparant avec les principaux langages que sont C++, Java, C# : syntaxe et concepts objet (classes, héritage, instanciation), gestion de la mémoire, chaînes de caractères, exceptions, multithreading, concept des propriétés, mécanismes de modifications à l'exécution... sans oublier les nouveautés d'Objective-C 2.0.

Critique du livre par Cyril Doillon

Pierre Y. Chatelier, auteur de l'ouvrage "Objective-C pour le développeur avancé" ne vous est très certainement pas inconnu, puisqu'il est l'auteur de trois cours consacrés l'un à Mac OS X 10.4 ([Lien110](#)), l'autre à Mac OS X 10.5 Leopard ([Lien111](#)) et le troisième à Objective-C ([Lien112](#)), cours hébergés sur <http://pierre-chatelier.developpez.com/> ([Lien113](#)).

Dans son ouvrage « Objective-C pour le développeur avancé », il propose d'initier le lecteur à la programmation pour iPad, iPhone et Mac OS X par l'intermédiaire du langage "Objective-C".

Il a choisi de le mettre en parallèle avec les langages actuels les plus courants, à savoir Java, C++ et C#.

Le livre commence ainsi par une présentation globale du langage pour se terminer par les spécificités du langage d'Apple.

Ici, l'auteur met en parallèle, principalement au début de l'ouvrage, les spécificités et bases des langages C++, Java et C# avec les particularités de l'Objective-C, autant sur la syntaxe que dans le comportement.

Cela permet ainsi, aux développeurs ayant leurs marques dans ces langages d'aborder le développement en Objective-C plus aisément.

Plus la lecture avance, plus les fonctionnalités spécifiques du langage d'Apple sont présentes et plus ce parallèle avec les autres langages disparaît.

Cela donne une très bonne progression dans l'apprentissage du développement en Objective-C à partir de bases connues.

Du côté de la forme, le livre se présente d'une manière assez classique avec un chapitrage découpé par grands ensembles de la programmation objet orienté sur les spécificités de l'Objective-C.

On retrouve, dans chaque chapitre, des explications claires sur chaque point abordé avec des exemples efficaces.

On notera particulièrement que certains exemples mettent en évidence un code en langage C++, Java ou C# avec son équivalent Objective-C placé juste à côté.

Ce dernier point est très important pour que les développeurs fassent un comparatif rapide et acquièrent les bases nécessaires.

Autant sur le fond que sur la forme, le livre de Pierre Y. Chatelier, "Objective-C pour le développeur avancé" propose une approche intéressante pour le passage des langages C++, Java et C# vers le langage d'Apple. Le seul point noir que l'on peut remarquer est justement que des novices des langages de base pourront être assez vite perdus car des bases de la programmation objet ne sont que très vite abordées. Mais l'objectif étant d'apprendre l'Objective-C aux développeurs avancés, celui-ci peut être considéré comme atteint.

Le livre présentant également des fonctionnalités avancées du langage, il permettra de se lancer assez facilement dans la programmation sur iPhone, iPad et Mac OS X.

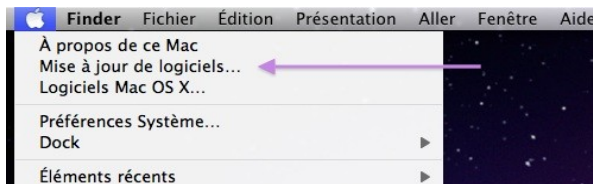
Retrouvez cette critique de livre sur la page livres Mac : [Lien114](#)

Installation des Developer Tools

C'est parti pour le premier tutoriel de la section iOS. Et comme il ne s'agit pas de perdre en route les bonnes habitudes de la maison, je vous propose de faire le point sur les outils qui vous seront utiles/indispensables pour le développement de votre première application iPhone.

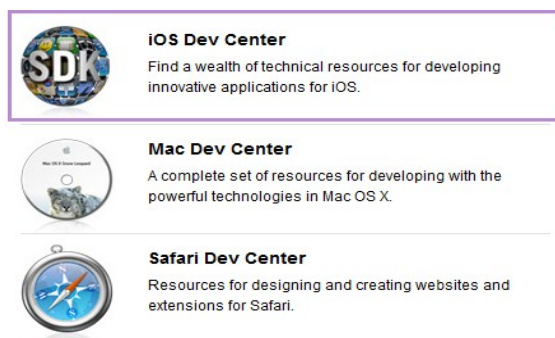
1. MacOS X : le passage obligé

Pour développer une application pour l'iPhone, la tester sur votre iPhone, la soumettre à Apple, il vous faut un Mac. Pour être plus précis, il vous faut un ordinateur avec le système d'exploitation MacOS X Snow Leopard. Pour que l'installation fonctionne, il est fortement recommandé d'avoir validé toutes les mises à jour de logiciels proposées par Apple (Menu Pomme > Mise à jour de logiciels...)



2. Découvrez l'iOS Dev Center

Apple met à la disposition des développeurs Mac et iPhone un site Internet dédié rempli de documents, vidéos, procédures, ressources, aides, ou exemples de code. Ce site Internet est divisé en trois « Dev Centers » : pour le Mac, pour l'iPhone et l'iPad, et le petit dernier pour Safari, le navigateur Internet. Pour vous y rendre, une adresse : <http://developer.apple.com> ([Lien115](#)) puis cliquez sur iOS Dev Center



De nombreuses ressources vous sont alors proposées. Si ce n'est pas l'objet du présent article de toutes les détailler (ce serait d'ailleurs probablement infaisable), je vous encourage à y passer tout de suite un peu de temps, prendre vos marques : pour peu que vous soyez assez à l'aise avec l'anglais, cet espace peut s'avérer être une vraie mine d'or.

3. Créez votre compte de développeur Apple

Pour pouvoir aller plus loin, vous devez vous enregistrer comme développeur Apple. Sans cela vous ne pourrez pas

accéder aux ressources du Dev Center et télécharger les outils qui vous permettront de développer. Cette inscription est gratuite, sans engagement particulier et ne prend que quelques minutes. Les liens d'accès sont disponibles dans le coin supérieur droit de quasi tous les écrans.



Vous avez par ailleurs la possibilité de vous enregistrer avec votre Apple ID (celui-là même qui vous sert à télécharger les derniers jeux pour votre iPhone).

4. Téléchargez le dernier SDK

Une fois connecté, sur la page d'accueil de l'iOS Dev Center, une zone vous propose de télécharger le dernier SDK disponible avec XCode. En effet, Apple associe quasi systématiquement le Kit de Développement Logiciel avec les dernières versions de tous les outils nécessaires.

Conseil important : Apple propose une nouvelle version du SDK pour chaque mise à jour importante d'iOS. Par ailleurs, pour permettre aux développeurs de proposer des applications tenant compte des futures évolutions, des SDK bêta peuvent aussi être proposés. Privilégiez systématiquement le dernier SDK stable. Autrement dit, il est inutile de télécharger la version bêta : vous avez besoin d'un environnement stable et propre. Vous allez probablement avoir assez de mal à déboguer vos applications pour ne pas ajouter les bogues des autres !

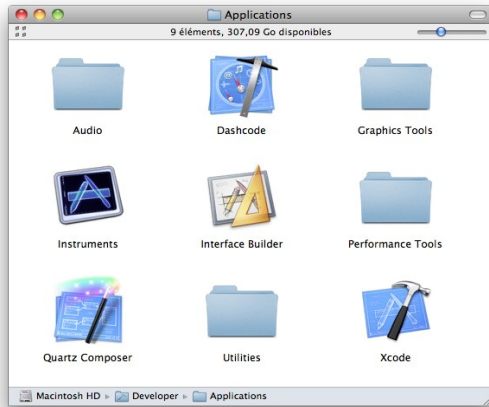
Cliquez et... faites chauffer la ligne : quelques gigaoctets à télécharger quand même. Vous avez probablement le temps d'aller faire quelques courses.

5. Installez

Une fois le téléchargement achevé, double-cliquez sur le fichier DMG pour monter l'image disque. Double-cliquez sur l'icône d'installation pour commencer la procédure et laissez-vous guider. Je ne vous fais pas l'affront de détailler les boutons « Continuer » et « J'accepte » qui vous attendent dans cet installeur. Notez cependant que dans la rubrique « Type d'installation », les cases cochées par défaut sont très bien : autant les laisser dans cet état (tout coché, sauf « Mac OS X 10.4 SDK ») et passer à la suite

6. Faites le tour du propriétaire

Les applications de développement sont installées par défaut dans un dossier **Developer** à la racine de votre disque principal. Il contient en outre différents documents, outils, éléments de référence et exemples de code. Mais le plus intéressant est le sous-dossier **Applications** qui contient les programmes qui vous serviront pour le développement iOS :



Xcode est l'application centrale du dispositif : gestionnaire

Les livres iOS

Développer pour l'iPhone et l'iPad - Le guide du SDK. Créez vos applications pour l'App Store

Ce livre s'adresse à tous les développeurs qui souhaitent se lancer ou se perfectionner dans la création d'applications pour les périphériques tactiles Apple et leur mise en ligne sur la plateforme App Store.

Cet ouvrage est conçu pour couvrir, de manière unifiée et progressive, l'ensemble du cycle de développement et expliquer les notions qui sont spécifiques à cette plateforme. Il permettra de construire avec rigueur et élégance des « applis » pour tous les terminaux de la famille iPhone / iPad.

Les fonctions mises en valeur et les exemples abordés suivent les préconisations et les modèles de programmation des ingénieurs Apple. Le lecteur dispose ainsi de bases solides pour répondre aux exigences de qualité et de distribution.

Cet ouvrage couvre les API introduites à partir des versions 3.1 et 3.2 du SDK avec, en particulier, les nouvelles classes de reconnaissance gestuelle (UIGestureRecognizer), les contrôleurs de vues pour l'iPad (*popover* et *splitview*) le support de la réalité augmentée, et beaucoup d'autres nouveautés qui multiplient les possibilités de l'iPhone ou de l'iPad.

Pour que ce livre soit un outil toujours à jour, il est prolongé par un site web qui l'actualise en permanence : [Lien117](#).

Une interview de l'auteur est également disponible sur le site de Dunod à l'adresse suivante : [Lien118](#)

de projets, il vous permet d'éditer les codes sources, prend en charge la compilation et encadre l'exécution des applis que vous développez ;

- **Interface Builder** vous permettra de modifier vos vues, de mettre en place votre interface utilisateur ;
- **Instruments** vous permettra de tracer les pertes mémoire, et d'améliorer la performance de vos applications ;
- **Dashcode** est un éditeur HTML CSS permettant de faire des widgets Dashboard ou des sites Internet dynamiques ;
- **Quartz Composer** vous permet de créer des animations graphiques à intégrer dans les applications MacOS sans écrire une ligne de code.

Cette fois c'est bon, vous êtes prêt pour le développement iPhone ! Les tutoriels suivants vous proposeront de créer vos premières applications iPhone, et de vous familiariser avec Xcode.

N'hésitez pas à nous faire part de vos remarques dans les commentaires : ça nous encouragera pour la suite ! Merci !

Retrouvez l'article d'Axon en ligne : [Lien116](#)

Critique du livre par Yan Verdavaine

Quand j'imagine ce que devrait être un livre informatique, ce livre est pour moi l'une des meilleures réponses. Autant accessible aux débutants qu'aux expérimentés, ce livre est une mine d'or quand on veut commencer ou approfondir le développement d'iOS.

Etienne Vautherin a travaillé pendant 12 années au support technique et au marketing technologique d'Apple auprès des développeurs. C'est une personne qui a l'habitude d'expliquer et d'aider les développeurs.

Et cela se voit tout au long du livre. Dans toutes ses explications, il va à l'essentiel. Et si vous voulez plus de précision, il n'hésite pas à donner des références d'Apple ou tutoriels tiers disponibles sur Internet gratuitement.

Vous y trouverez, par exemple, cité le tutoriel "De C++ à Objective-C" de Pierre Chatelier ([Lien119](#)).

Une grande qualité de ce livre est son plan. Vous commencerez par une présentation de Xcode et du débogueur suivie par une présentation des fondations du framework. Une petite étape sur l'écosystème et vous arriverez sur le modèle MVC proposé par iOS.

Vous terminerez par un vaste paysage des possibilités d'IHM et d'ergonomie que propose iOS.

Le dernier chapitre est bien sûr consacré à l'App Store.

Ce livre est pour moi une réussite. L'auteur a su ressortir l'essentiel de la plateforme iOS en seulement 400 pages.

Le livre terminé, vous n'hésitez pas à y revenir de temps en temps pour clarifier quelques concepts ou redécouvrir quelques possibilités.

Retrouvez cette critique de livre sur la page livres iOS : [Lien120](#)

Bases de données relationnelles et normalisation

La normalisation des tables (plus formellement relations) qui composent une base de données relationnelle est incontournable si l'on veut assurer à celle-ci :

- un contenu valide, par l'élimination de la redondance de l'information au sein de chaque table, redondance pouvant être la cause d'incohérences lors des opérations de mise à jour ;
- un niveau de qualité faisant qu'une évolution de la structure de cette base de données soit facilitée quand les règles du jeu de l'entreprise changent.

Certes, la normalisation ne résout pas tout, car certaines redondances mettant en jeu plus d'une table peuvent échapper à son contrôle, mais elle joue en tout cas un rôle décisif. De cela, nous avons pu en juger pendant de nombreuses années, sur le terrain, que les bases de données soient relationnelles ou non.

La normalisation est un sujet dont l'étude n'est pas toujours simple, car elle a fait l'objet d'une théorie élaborée par des mathématiciens, pourvoyeurs en l'occurrence de bien des théorèmes, dont certains incontournables pour s'assurer de la structure correcte des données. L'objet de cet article est de tenter de rendre le sujet abordable, l'expliquer sans employer pour autant le langage parfois hermétique de nos chers théoriciens. Les informaticiens ont pris le relais et le sujet — fort en vogue dans les années soixante-quinze / quatre-vingt — fit l'objet d'une vulgarisation souvent douteuse au plan de la pertinence. Aussi, nous ne pouvons ni ne voulons nous contenter des définitions trop simples, voire fantaisistes ou fausses — touchant notamment aux formes normales — qui abondent dans de nombreux ouvrages ou sur la Toile : rigueur et pertinence sont de mise, il y va de la validité et de la crédibilité de la base de données.

Suite aux discussions auxquelles nous avons participé sur les forums Developpez.com, nous avons tiré un certain nombre d'enseignements quant aux questions que se posent certains qui découvrent la normalisation, ou d'autres qui ne l'ont pas abordée par le bon côté. Nous tentons ici de répondre au mieux à leurs attentes tout en formalisant un peu plus et en les creusant, certains points seulement effleurés. Cet article pourra évoluer en fonction de leurs remarques.

Bonne lecture.

N.B. Cet article traite des formes normales, de la première à la sixième ; il a été découpé en parties qui seront publiées progressivement.

1. Quelques remarques préliminaires

1.1. A propos de Tutorial D

Le thème de la normalisation est traité ici dans le cadre du Modèle Relationnel de Données, inventé par Ted Codd et qui continue à évoluer sous la houlette de ses continuateurs, citons Chris Date, compagnon de route de toujours de Ted, Hugh Darwen, David McGoveran. Le Modèle Relationnel a de nombreux points en commun avec le Modèle SQL qui s'en est inspiré, mais il y a des différences très sensibles tant sur le fond que sur la forme. Concernant la structure des données, ces différences seront

expliquées au fur et à mesure.

On trouvera en annexe (paragraphe A) la définition des concepts structurels fondamentaux propres au Modèle Relationnel et quelques notes concernant le langage **Tutorial D**, prototype relationnellement complet (cf. [Date 2006]). On y trouvera aussi (paragraphe B) quelques notes relatives à l'algèbre relationnelle, permettant de comprendre par exemple comment les opérations relationnelles sont exprimées en Tutorial D, avec au besoin, la traduction en SQL de ces opérations.

1.2. A qui s'adresse cet article ?

S'il n'est évidemment pas nécessaire que le lecteur sache tout du Modèle Relationnel, il est néanmoins préférable qu'il ait un minimum de connaissances en SQL :

Savoir ce qu'est une table (aspect structurel), comment on l'exploite (utilisation des opérateurs relationnels figurant plus ou moins explicitement dans le bloc SELECT-FROM-WHERE), et comment on garantit un strict minimum d'intégrité des données (clés primaires et étrangères).

Mais même s'il ne connaît que très peu SQL, celui qui construit des diagrammes au niveau conceptuel (MCD selon l'approche Merise ou Entité/Relation, IEF, voire diagramme de classes), est très concerné, puisque les tables qui constitueront la base de données seront le résultat de la dérivation de ce qu'il aura construit.

La normalisation permettra de s'assurer que les fondations de la base de données sont saines et robustes.

Tout en étant incontournable, la normalisation peut être considérée comme un sujet indépendant (orthogonal comme dirait Date), au point que Codd — qui connaît mieux que quiconque la chose — n'en parle dans son ouvrage de référence [Codd 1990] que pour rappeler l'intérêt qu'il y a à normaliser afin d'éviter les anomalies de mise à jour (deux pages et c'est tout). Quoi qu'il en soit, j'ai constaté que nombreux sont les « spécialistes » qui traitent du sujet de façon imparfaite, n'ayant vraisemblablement pas eu l'occasion — sinon la curiosité — d'en mesurer les effets bien concrets et préjudiciables dans le contexte de la Production informatique.

Cela dit, l'objet de l'article est quelque part de faire en sorte que le lecteur intéressé dispose de définitions exactes des formes normales, pour mieux bétonner sa base de données. J'essaie en l'occurrence de rester lisible car il est facile d'écrire de façon très hermétique sur le sujet, mais là n'est certainement pas le but de la manoeuvre.

Le thème de l'« optimisation » des bases de données est aussi abordé, mais on sort alors carrément du cadre de la théorie relationnelle, ce qui fait que tous les administrateurs de bases de données (DBA) ne seront pas forcément convaincus par mon approche — basée quand même sur quarante années d'exercice du métier, dans les secteurs d'activité les plus variés, avec tous les types de SGBD, dans le contexte des très grandes bases de données et le plus souvent au fond de la soute — (cf. le paragraphe F en annexe, dont la rédaction fut provoquée par une réflexion sur ce thème de l'optimisation, en réaction à la confusion faite avec celui de la dénormalisation, confusion entretenue (sans doute bien involontairement) par certains auteurs et « experts » (se reporter au paragraphe 1.7, notamment à ce qui a trait à l'identification relative)).

Périodiquement, on relève chez Developpez.com (Mâtin, quel site !) des questions posées par des étudiants, questions ayant trait à la normalisation, mais sur des aspects plutôt théoriques et qui ne concernent que modérément les praticiens. A leur intention, quelques réponses sont fournies en annexe (cf. paragraphes E.6 et E.7).

1.3. Concernant le vocabulaire

Lorsqu'il a inventé le Modèle Relationnel de Données en 1969, l'objet de l'étude de Codd était les relations. En 1974-1976, en créant SEQUEL (ou SQL si vous préférez), Chamberlin a préféré remplacer « relation » par « table » (mot sans doute plus parlant). Mais il y a toujours intérêt à rechercher le mot le plus approprié pour un concept, par souci justement de précision. Ainsi, en 1994, Date et Darwen (D & D, que j'appelle encore affectueusement les Dupondt) en sont arrivés à définir le concept de **variable relationnelle** (*relation variable*, en abrégé *relvar*). Ils ont estimé — à l'instar des autres théoriciens du relationnel, du reste — qu'une relation étant une **valeur**, elle ne se situe ni dans le temps ni dans l'espace, contrairement à la variable relationnelle, à laquelle on affecte, par le biais d'une requête (disons au sein d'un programme, d'une procédure ou de ce que vous voulez), des valeurs différentes (des relations) au fil du temps. Ainsi en va-t-il dans un programme, quand à la variable x on affecte les valeurs 1 ou 2, etc. qui, elles non plus, ne se situent ni dans le temps ni dans l'espace et dont nous ne représentons que des images.

Au contraire, en SQL on ne dispose que d'un concept, celui de table. Mais, quand une table est-elle une variable ? Une valeur ? L'ambiguïté peut devenir fort gênante.

J'aurais pu faire l'économie du terme *relvar* et me contenter de l'expression *schéma de relation* (comme l'on pratiquait du reste dans les années soixante-dix, quatre-vingt), mais en réalité un tel schéma (*en-tête* selon le Modèle Relationnel de Données) est en fait un des composants de la *relvar*, ça n'est qu'un ensemble d'attributs (au sens de la théorie des ensembles). Bref, quand on y a pris goût, on ne peut plus se passer de la *relvar*...

On pourrait encore poser la question : Mais pourquoi traiter de la normalisation en faisant appel au vocabulaire de Tutorial D (qui est celui du Modèle Relationnel) plutôt qu'à celui de SQL ? Je répondrai encore que c'est au nom de la précision dans la définition des concepts mis en jeu.

Par exemple, quand vous aborderez la forme normale de Boyce-Codd (BCNF), vous lirez ceci :

Une *relvar* R est en forme normale de Boyce-Codd (BCNF) si et seulement si les seules dépendances fonctionnelles non triviales qu'elle doit vérifier sont de la forme $X \rightarrow Y$, où X représente une surclé et Y un sous-ensemble d'attributs de l'en-tête de R .

C'est du béton, même si à ce stade vous ne pouvez peut-être pas encore en juger. A la sauce SQL, on pourrait reformuler cela à peu près ainsi :

Un schéma de table T^* est en forme normale de Boyce-Codd (BCNF) si et seulement si les seules dépendances fonctionnelles non triviales auxquelles il satisfait sont de la forme $X \rightarrow Y$, où X contient une clé candidate et Y représente un sous-ensemble de colonnes de T^* .

Je ne suis pas sûr qu'il n'y aurait pas de grincheux pour dire que ça n'est pas une très bonne formulation, que cela manque de rigueur (par exemple, quel sens exact prend le verbe contenir dans l'expression « contient une clé

candidate » ?)

Quoi qu'il en soit, je trouve plus difficile de dire les choses en termes SQL, aussi Tutorial D mérite-t-il qu'on s'y intéresse, même s'il ne s'agit pas ici de l'étudier pour lui-même.

Veuillez prendre note que je ne fournis pas toujours le texte original de certaines citations mais seulement leur traduction. On n'est jamais à l'abri d'une bévue, mais je pense en avoir respecté l'esprit.

2. De la normalisation

2.1. Le contexte

Quand on parle de bases de données relationnelles, on évoque inmanquablement les trois piliers qui constituent les fondements de la théorie relationnelle et ayant pour objet :

1. La **structure des données**, c'est-à-dire, si l'on se situe au niveau SQL, les règles de définition des tables en termes de lignes et de colonnes ;
2. La **manipulation des données** : comment exploiter ces tables, à l'aide par exemple — toujours dans le contexte SQL — de l'incontournable triplet SELECT, FROM, WHERE et des opérateurs INSERT, etc. ;
3. L'**intégrité des données**, c'est-à-dire les moyens mis à notre disposition par le SGBD, concourant à la validité de ces données, tels que les clés primaires, clés étrangères, assertions, triggers et contraintes diverses.

Il existe par ailleurs un volet extrêmement important concernant les bases de données relationnelles, celui de la **normalisation**, dont l'objet est double :

- à l'intersection d'une ligne et d'une colonne, certes on trouve des données de type très simple, telles que les habituels nombres et chaînes de caractères, mais peut-on aussi légalement trouver des données de types plus complexes, telles que des listes, des tableaux, des tables, etc. ? La normalisation a pour objet de définir les règles du jeu à ce sujet, en relation avec les effets que cela peut avoir sur chacun des trois piliers précédents.

Cela concerne au premier chef ce qu'il est convenu d'appeler la Première Forme Normale (1NF) et sera développé dans le paragraphe 2 « Première forme normale » ;

- la normalisation a aussi pour objet de fournir les outils et les techniques nous permettant de débusquer, d'éliminer les **redondances** qui non seulement rendent les tables obèses, mais par ailleurs nous compliquent la vie lors des opérations qui mettent à jour celles-ci (mises à jour nécessairement redondantes elles aussi) finissant par rendre faux le contenu de la base de données, sans parler de l'effet néfaste sur les performances. Par voie de conséquence, en normalisant, tout en éliminant ce genre d'*impedimenta*, on améliore l'architecture de la base de données, ce qui n'est pas un mince avantage, ne serait-ce que si l'on a un jour à faire

évoluer celle-ci.

Cela concerne les formes normales suivantes (2NF à 5NF) et sera développé dans les paragraphes 3 et suivants. Vu sa spécificité (garde-fou dans la manipulation des données intervallaires), la 6NF sera traitée dans le contexte de la modélisation des données temporelles.

L'objet de la normalisation est repris dans le paragraphe 1.4.

2.2. Retour aux sources

Tout d'abord, il sera régulièrement fait ici référence à Ted Codd (1923-2003), le génial inventeur du Modèle Relationnel de Données, qui a tout de suite traité de la normalisation, de manière très rigoureuse.

Ensuite, bien que Ted Codd ne traite que des **relations**, suivant le contexte, on utilisera aussi par la suite les termes « **table** » et « **relvar** » (voir ci-dessous : « Relvar, relation et table »). En attendant, commençons par rappeler ce qu'est une relation dans le cadre de la théorie relationnelle.

Alors que SQL n'était pas encore né, et pour cause — et *a fortiori* le concept SQL de table — voici la définition donnée par Ted Codd de la relation dans son article fondateur (cf. [Codd 1970], paragraphe 1.3) :

« Le terme *relation* est utilisé ici dans son acception mathématique. Étant donnés les ensembles S_1, S_2, \dots, S_n (non nécessairement distincts), R est une relation sur ces n ensembles si c'est un ensemble de n -uplets, le 1er élément de chacun d'eux tirant sa valeur de S_1 , le 2e de S_2 , et ainsi de suite (de manière plus concise, R est un sous-ensemble du produit cartésien $S_1 \times S_2 \times \dots \times S_n$). On fera référence à S_j comme étant le j ème domaine de R . Suite à ce qui vient d'être énoncé, on dit que R est de degré n . Les relations de degré 1 sont souvent dites unaires, celles de degré 2 binaires, de degré 3 ternaires, et celles de degré n n -aires. »

Une représentation imagée d'un n -uplet d'une relation R de degré n :

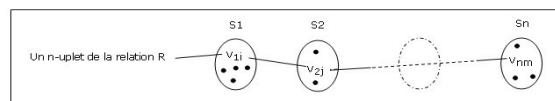


Figure 2.1 - Un n -uplet, comme des perles qu'on enfle

Codd poursuit :

« Pour simplifier l'exposé, on utilisera souvent une représentation des relations sous forme de tableaux ... Un tableau représentant une relation a les propriétés suivantes :

- (1) Chaque ligne représente un n -uplet de R ,
- (2) L'ordre des lignes n'a aucune importance,
- (3) Toutes les lignes sont distinctes,
- (4) L'ordre des colonnes est significatif - il correspond à l'ordre S_1, S_2, \dots, S_n des domaines sur lesquels sont définis les domaines de R .
- (5) La signification de chaque colonne est en partie rendue en l'affectant du nom du domaine correspondant. »

Représentation d'une relation R de degré n, sous forme imagée, rectangulaire, plate et traditionnelle :

R (S1	S2	...	Sn)
v11	v21	...	vn1
...
v1i	v2j	...	vnm
...

Figure 2.2 - Une relation sous forme de tableau

Mais attention, l'image n'est pas la chose !

Dans ce qu'a écrit Codd, un point important peut paraître aujourd'hui choquant : il est en effet précisé que l'**ordre des colonnes** est significatif (point 4). Cela vient du fait que, dans ce tout premier jus du Modèle Relationnel, une colonne n'a pas de nom en propre, elle hérite implicitement de celui de son domaine de référence : contexte mathématique oblige. Ce n'est qu'en 1971 (cf. [Codd 1971], page 31) qu'apparaît le concept d'attribut, débarrassant le Modèle Relationnel de cette fâcheuse contrainte. Je cite (en rappelant que l'avatar SQL de la relation est la *table* et que le *degré* d'une relation correspond au nombre de colonnes d'une table) :

« Les n domaines ne sont pas nécessairement distincts. Plutôt qu'utiliser un ordre pour déterminer chaque domaine référencé (comme cela se fait en mathématiques), on utilisera un nom distinct pour chaque référence faite et nous l'appellerons nom de l'attribut... En conséquence, chaque référence faite à un domaine lors de la définition de R est appelée attribut de R. Par exemple, une relation de degré 3 pourrait avoir pour attributs (A1, A2, A3) tandis que les domaines correspondants pourraient être les domaines (D5, D7, D5). Les noms d'attributs sont un moyen d'éviter d'imposer aux utilisateurs la connaissance de la position des domaines. »

2.3. Rappel de quelques définitions

Les définitions qui suivent reprennent celles de Chris Date (cf. le paragraphe A en annexe).

Un **domaine** tel que S_j est un ensemble de valeurs, par exemple celui des entiers, celui des chaînes de caractères, ceux des dates, des points, des lignes, des ellipses, polygones, des numéros de Siret, des codes postaux, des ISBN, des EAN13, etc. A noter qu'aujourd'hui on utilise le terme **type** plutôt que le terme domaine.

Un **n-uplet** (ou **tuple**) est une valeur. C'est un ensemble de triplets de la forme $\langle A_i, D_i, v_i \rangle$ où A_i désigne un **nom d'attribut**, D_i désigne un nom de domaine et v_i une valeur appartenant au domaine D_i . Le couple $\langle A_i, D_i \rangle$ est un **attribut** du n-uplet ; v_i est la **valeur d'attribut** de l'attribut A_i ; le domaine D_i en est le **domaine d'attribut** correspondant (**type d'attribut**).

Exemple graphique, plat : un n-uplet composé des triplets $\langle \text{Attr1}, D_1, a_{11} \rangle, \langle \text{Attr2}, D_2, a_{21} \rangle, \dots, \langle \text{Attrn}, D_n, a_{n1} \rangle$

Attr1 : D1	Attr2 : D2	...	Attrn : Dn
a11	a21	...	an1

Figure_2.3a - Un n-uplet au format tabulaire

Autre exemple plus parlant de n-uplet : un certain membre chez developpez.co

Membre : Char	Statut : Char	...	Localisation : Char
Antoine	expert	...	Paris

Figure 2.3b - Autre exemple de n-uplet au format tabulaire

Une **relation** est une valeur. Plus précisément, c'est une valeur constituée d'un **en-tête** (ou schéma ou intension, notez l'orthographe) et d'un **corps** (extension). L'en-tête est composé d'un ensemble d'**attributs**. Le corps est l'ensemble des **n-uplets** composant la relation.

Exemple de représentation sous forme tabulaire d'une relation n-dimensionnelle (tout en rappelant que l'image d'une chose n'est pas la chose) :

R	Membre : Char	Statut : Char	...	Localisation : Char	← En-tête de la relation R } Corps de la relation R
	Antoine	expert	...	Paris	
	Philou	confirmé	...	Atlanta	
	Frédéric	expert	...	Paca	

Figure 2.4 - Composants d'une relation

Dans un contexte informel, il est courant de ne pas faire figurer le nom des domaines dans [l'image de] l'en-tête :

R	Membre	Statut	...	Localisation
	Antoine	expert	...	Paris
	Philou	confirmé	...	Atlanta

	Frédéric	expert	...	Paca

Figure 2.5 - Représentation informelle d'une relation

Relvar, relation et table

Au vu de ces représentations, on pourrait inférer qu'une **table SQL** est une relation (en remplaçant les termes « attribut » et « n-uplet » respectivement par « **colonne** » et « **ligne** »). Ça n'est pas exactement le cas, car (outre bon nombre de propriétés non nécessairement partagées) une table peut changer de valeur, tandis qu'une relation **est** une valeur, donc par définition invariable, tout comme les entiers 1 ou 2. L'aspect variable des choses concerne la **variable relationnelle** (en abrégé **relvar**), type de variable affectée successivement de valeurs qui sont des relations : une relation y remplace une autre lors d'une opération de mise à jour. Notons que Codd n'utilisait pas le terme relvar, mais l'expression « time-varying relation », qui n'est plus jugée pertinente aujourd'hui, du fait du caractère justement invariable des relations.

Et n'oublions pas qu'une table SQL peut n'être qu'un **sac** (*bag*), dans la mesure où la présence d'une clé (disons primaire) n'est pas exigée, ce qui autorise l'existence de lignes en double (or un sac n'est pas un ensemble).

2.4. Objet de la normalisation

a) A propos de la première forme normale (dont l'étude sera développée dans le paragraphe 2).

Les relations sont des êtres mathématiques. Elles sont soumises à certaines contraintes structurelles et leur finalité est d'être manipulées, combinées, à l'aide de l'**algèbre relationnelle** ou du **calcul relationnel** (qui est une application du calcul des prédicats). Ayant une

préférence pour le calcul des prédicats, Ted Codd a raisonné en logicien. Nous verrons à l'occasion de l'étude de la première forme normale, qu'en 1969, il se plaça d'entrée dans le cadre de la logique du deuxième ordre, jugeant l'année suivante que la logique du premier ordre suffisait pour manipuler les relations. L'adéquation du calcul relationnel (et par contrecoup de l'algèbre relationnelle) à la logique du premier ordre eut pour conséquence une contrainte forte, conduisant à normaliser les relations en ce qu'il est convenu d'appeler la première forme normale (1NF), selon laquelle une relation ne peut pas être une valeur pour un attribut d'une autre relation : par exemple, les lignes de facture d'une facture ne peuvent pas être des valeurs d'un attribut LigneDeFacture d'une relation Facture.

Certes, avec des systèmes comme IMS/DL1, par construction (modèle hiérarchique oblige), les lignes de facture sont nichées dans les factures, les engagements sur lignes de facture sont nichés dans les lignes de facture, etc. Mais IMS/DL1 ne permet pas de manipuler des **ensembles** à l'aide d'une algèbre ou d'un calcul, on est à un niveau inférieur où l'on ne traite qu'un **enregistrement** à la fois et, dans ces conditions, il n'y a évidemment aucune contrainte sur la façon de structurer les données.

b) A propos des autres formes normales (dont l'étude sera développée dans les paragraphes 3 et suivants).

Ce que l'on appelle deuxième forme normale, troisième forme normale et forme normale de Boyce-Codd sont les éléments d'une théorie, d'abord développée par Codd dès 1970, puis complétée par Raymond Boyce (prématurément disparu en 1974). Huit ans après que Codd l'ait eu entamée, des mathématiciens comme Jorma Rissanen et Ronald Fagin prirent le relais et la théorie de la normalisation fut achevée en 1979, avec la mise à notre disposition des quatrième et cinquième formes normales.

Pour reprendre ce qui a été évoqué au paragraphe 1.1, respecter ces formes normales a pour effet (entre autres choses) de débarrasser les relations de **redondances** non seulement inutiles et causes d'obésité, mais surtout génératrices d'erreurs eu égard aux règles de gestion des données de l'entreprise, lors des opérations de **mise à jour** (disons INSERT, UPDATE, DELETE). Ces redondances sont le plus souvent la conséquence d'une modélisation conceptuelle en amont maladroite, voire inexistante, ou encore le mauvais fruit d'une « dénormalisation » inopportune.

c) Observations concernant la modélisation conceptuelle.

Lorsqu'on représente les données sous forme graphique : modèles conceptuels de données (MCD) de la méthode Merise, et plus généralement diagrammes entités/relation (voire diagrammes de classes), il y a tout un travail de vérification concernant chaque type d'association (ce qu'on désigne encore par association-type ou relation-type) entre types d'entités, consistant à « s'assurer que chacune des propriétés ne peut être vérifiée sur un sous-ensemble de la collection de la relation-type » [TRC 1989]. Attention, dans cette citation, la relation-type n'a rien à voir avec la relation du Modèle Relationnel, il s'agit de l'association (*relationship*) existant entre entités-types. Ce travail de

vérification — portant lui aussi le nom de normalisation — conduit à expulser au besoin une propriété d'une association-type vers une entité-type (ou inversement). Ceci a à voir avec ce que Codd appelle la normalisation en deuxième forme normale (2NF), laquelle a en vérité une portée bien plus étendue, car elle concerne l'ensemble des relvars composant une base de données relationnelle. La 2NF est aussi beaucoup plus formelle quant à son énoncé.

La normalisation joue un rôle crucial quant à la qualité de l'architecture de la base de données, laquelle doit être structurellement valide et apte à évoluer, premièrement par le recours à une démarche au niveau conceptuel **synthétique, descendante** (donc en amont), à l'aide par exemple de la méthode Merise (démarche valant également pour les diagrammes de classes), deuxièmement par une vérification rigoureuse, mettant en jeu une démarche **analytique, ascendante**, pour laquelle on s'appuie justement sur la théorie de la normalisation : l'architecture de la base de données relève ainsi d'une approche mixte où l'on pratique l'art du yoyo, en alternant les deux démarches.

d) Prise en compte des données temporelles.

Ensemble, Hugh Darwen, Nikos Lorentzos et Chris Date, compagnon de route, fils spirituel (et parfois rebelle) de Ted Codd, ont enrichi la théorie relationnelle, en approfondissant avec une extrême rigueur le domaine des bases de données **temporelles**, et en nous fournissant les techniques pour nous y lancer à notre tour, autrement qu'à l'instinct, comme c'est hélas trop souvent le cas, ou sur la base de travaux théoriques jugés défectueux (cas de TSQL2 qui fut proposé pour être intégré à SQL/2). C'est un sujet d'étude essentiel, que tous les concepteurs devraient approfondir, car depuis toujours, la prise en compte du temps dans les bases de données a été, et reste quelque chose de compliqué et d'omniprésent, au moins dans le monde mouvant et agité de l'assurance, de la banque, de la grande distribution, de la retraite, et j'en passe.

Maintenant, comme dit Date : « Normalization is **no panacea** but it's a lot better than the alternative! »

2.5. Les étapes de la normalisation.

L'usage veut que l'on normalise en procédant par étapes : dans un premier temps, on s'assure que l'on respecte ce que l'on appelle la première forme normale (1FN ou 1NF) déjà évoquée et qui a à voir avec le typage des attributs, à savoir qu'une relation ne peut pas être (en principe) une valeur pour un attribut d'une autre relation. Par exemple, les lignes de facture d'une facture ne peuvent pas être (en principe) des valeurs d'un attribut LigneDeFacture d'une relation Facture.

Ensuite, on s'assure que chaque relvar (ou table dans le contexte SQL), respecte ce que l'on appelle la 2e forme normale (2FN ou 2NF), puis la 3e forme normale (3FN ou 3NF), la forme normale de Boyce/Codd (FNBC ou BCNF), la 4e forme normale (4FN ou 4NF), la 5e forme normale (5FN ou 5NF) encore appelée PJ/NF (Project/Join Normal Form, forme normale par projection/jointure). La **projection** et la **jointure naturelle** (cf. le paragraphe B en

annexe) sont les deux opérations utilisées tout au long du processus (d'où l'expression « normalisation par projection/jointure »). La projection est utilisée pour remplacer une relvar R qui ne respecte pas la xNF par deux relvars R1 et R2 qui la respectent, et la jointure naturelle est utilisée pour retrouver très exactement R à partir de R1 et R2 au cas où le besoin s'en ferait sentir (par exemple au moyen d'une **vue**, ce qui permet de respecter le principe de l'**indépendance logique des données**, en garantissant une simplification de la manipulation des données par l'utilisateur, une stabilité de la représentation de celles-ci, tout à fait profitable pour les applications, etc.). Pour les données temporelles (plus généralement intervallaires), il est recommandé de pousser jusqu'à la 6e forme normale (6FN ou 6NF), qui marque la fin du processus de normalisation.

Il va sans dire que la normalisation par projection / jointure préserve l'information, c'est-à-dire le contenu de la base de données.

On notera qu'une relvar en 2NF est nécessairement en 1NF, qu'une relvar en 3NF est nécessairement en 2NF, etc., d'où la traditionnelle représentation graphique (enrichie de la 6NF) :

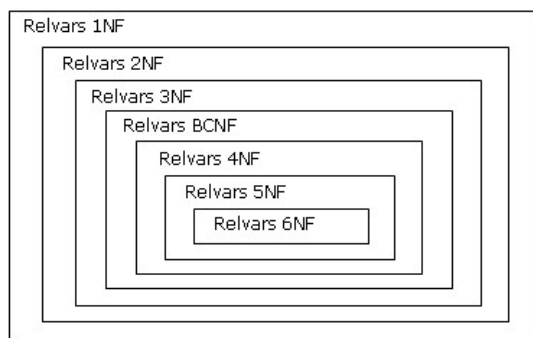


Figure 2.6 - Les relvars à la manière des matriochkas

Quand on a acquis un certain entraînement, on peut s'intéresser directement à la BCNF et se dispenser des étapes consistant à s'assurer que l'on est en 2NF et 3NF. Quant à la 4NF et à la 5NF, la partie est réputée assez difficile et l'on fait souvent l'impasse, en espérant que le MCD (modèle conceptuel de données) ou le diagramme de classes que l'on a réalisés soient normalisés, ce qui est en principe le cas avec des concepteurs expérimentés, mais attention quand même aux surprises. Quant aux Sotomayor de la normalisation, ils se dispensent des barres intermédiaires et se mettent en condition en attaquant tout de suite la barre la plus haute.

2.6. Normaliser, une obligation ?

Pour être conforme au Modèle Relationnel, chaque relvar d'une base de données doit **nécessairement** respecter la 1NF (sinon ça n'est pas une relvar). Normaliser en 2NF et au-delà, est très vivement **recommandé**, même si d'un point de vue théorique ça n'est pas une obligation stricte, dans la mesure où l'algèbre relationnelle n'en subit pas les effets.

Mais nombreux sont ceux qui, sous l'emprise de l'émotion ou par crédulité (lecture de la presse du cœur

informatique, ragots de cafétéria, influence des légendes en tous genres colportées depuis l'arrivée des premiers SGBDR ...), préconisent une **non-normalisation a priori** et se limitent au respect de la 1NF. En effet, la normalisation conduit à casser une relvar en deux ou plusieurs relvars, en conséquence de quoi, « dénormaliser » (réassembler les morceaux) serait synonyme d'**optimiser**. Ceux-là devraient avoir à l'esprit cette réflexion de Donald Knuth ([Lien121](#)) (qui l'a peut-être empruntée à Tony Hoare) : « Premature optimization is the root of all evil. » Quelques variantes des arguments avancés par les ignorants :

A cause de la **jointure**, les requêtes deviendraient compliquées et Developpez.com abonde en commentaires du genre : « D'accord, je vais faire comme vous dites, même si ça complique un peu les requêtes ». Mais le DBA se fera un plaisir d'encapsuler ces requêtes dans des vues, lesquelles seront pour l'utilisateur des relvars comme les autres (respect du principe de l'**indépendance logique**).

Recomposer une relvar à partir des morceaux fait une fois de plus intervenir l'opération de **jointure**, que l'on qualifie hâtivement de non performante, en toute méconnaissance de cause, c'est-à-dire en confondant allègrement le niveau logique et le niveau physique. Certes, selon un raisonnement simpliste et en se plaçant au niveau physique, si les enregistrements impliqués ne sont pas dans la même page (bloc physique) sur le disque, la jointure serait source d'accès au disque (ou au cache) supplémentaires, mais son statut d'opération relationnelle par excellence fait qu'elle est l'objet de tous les soins de l'optimiseur des SGBD relationnels dignes de ce nom et qu'il n'y a pas lieu de s'affoler. Se reporter à ce sujet au paragraphe 3.8. La jointure a bon dos, les problèmes de performance sont ailleurs.

Dénormaliser n'est pas interdit, mais on en connaît aussi les inconvénients, par exemple :

Dégradation de la qualité de la modélisation que l'on finit par ne plus maîtriser et faire évoluer proprement.

Nécessité de mettre en œuvre des contraintes (assertions ou triggers SQL) garantissant sous le capot le respect de certaines dépendances fonctionnelles, multivaluées, etc. qui sont les conséquences du non respect de la normalisation.

Anomalies potentielles de mise à jour, obésité des tables SQL due à l'inflation des redondances, à leur enneigement (données utiles clairsemées, noyées au milieu de masses de nulls — lesquels ne facilitent pas la vie des optimiseurs —, et de valeurs par défaut plus ou moins pertinentes).

Parce qu'un SGBD comme DB2 ne respecte qu'en partie le principe de l'**indépendance physique** (à chacun de voir ce qu'il en est quant à son SGBD favori), un tuple d'une relation (ligne d'une table) est (physiquement) logé en totalité au sein d'un enregistrement sur le disque, c'est sommaire, mais c'est ainsi. En conséquence, l'accès à une valeur d'attribut d'un tuple provoque l'accès à l'ensemble des valeurs d'attributs de ce tuple. Suite à dénormalisation, la taille d'un enregistrement physique est supérieure à celle

des enregistrements « normalisés », ce qui fait qu'il y aura moins d'enregistrements par page (bloc physique sur le disque). Ainsi, pour un traitement séquentiel par lots (batch) ou pour des transactions lourdes, le nombre de lectures/écritures sera accru et la durée du traitement en pâtira d'autant.

Les opérations de mise à jour ne sont pas à l'abri : là encore, du fait du non-respect de l'indépendance physique, la mémoire sera inutilement encombrée et les temps de traitement pénalisés, car le SGBD manipulera des enregistrements pondéralement surchargés.

Il faut descendre dans la soude et **prouver** le bien-fondé de cette dénormalisation, résultats de mesures sérieuses en main, suite à des séances de **prototypage de performance** poussées. Un exemple simple, celui qui est considéré au paragraphe 3.8 est caractéristique de situations où l'on se trompe de cible, et dans lesquelles dénormaliser revient à appliquer un cautére sur une jambe de bois sans améliorer la performance. Dénormaliser devient dangereux (anomalies potentielles de mises à jour, coût du stockage), comme par exemple dans le cas de l'hypothétique table des membres de DVP (cf. paragraphes 3.1.2 et 3.1.3). Une réflexion attentive portant sur cette table (cf. Figure 3.2) incite à penser que c'est bien plus la dénormalisation que la normalisation qui peut être source d'une inflation de lectures/écritures physiques sur disque et d'encombrement des caches.

En passant : existe-t-il des règles logiques nous permettant de dénormaliser de façon rationnelle ? En ce sens, je traduis Chris Date ([Date 2007b], Chapitre 9, « Denormalization Considered Harmful » :

« Je voudrais mettre l'accent sur un point : une fois que l'on a décidé de dénormaliser, on s'est engagé sur une pente fort glissante. Question : Quand s'arrêter ? Avec la normalisation, la situation est différente car on a des raisons logiques et claires de poursuivre le processus jusqu'à ce qu'on ait atteint la forme normale la plus élevée possible. Doit-on en conclure qu'avec la dénormalisation on ait à procéder jusqu'à atteindre la forme normale la moins élevée qui soit ? Bien sûr que non, à ce jour nous ne disposons pas de critères logiques permettant de décider quand arrêter le processus. En d'autres termes, en choisissant de dénormaliser, on a décidé d'abandonner une position qui offre une base scientifique et une théorie logique solides, pour la remplacer par quelque chose de purement pragmatique et nécessairement subjectif. »

Au fond, la dénormalisation est un bel exemple d'*ignoratio elenchi*. Et comme le fait observer Frédéric Brouard (SQLpro) :

« On peut aussi pousser le bouchon à l'extrême : pourquoi pas une seule table contenant tout dans la base ? Vous commencez donc à douter de l'efficacité du tout au même endroit... Mais où il faut-il s'arrêter ? Où placer le curseur ? C'est justement l'art du respect des formes normales qui nous en donne la clef ! »

Signalons quand même une situation dans laquelle on peut cette fois-ci se poser légitimement la question de la dénormalisation d'une relvar : il s'agit de la situation dans

laquelle la BCNF serait violée, alors que la normalisation n'arrangerait pas forcément les choses. Ceci est abordé au paragraphe 3.7. Mais il faut reconnaître que cette situation ne se présente heureusement pas souvent. Sinon, si la modélisation conceptuelle des données est réalisée selon les règles de l'art, on n'a guère de raison objective de ne pas normaliser.

Ne pas normaliser est en général la conséquence d'une modélisation conceptuelle des données absente ou non maîtrisée.

Un dernier point. Si d'aucuns admettent que les bases de données utilisées dans un contexte transactionnel doivent être normalisées, ils n'ont aucun état d'âme à dénormaliser à tout va les tables de dimension dans le contexte des bases de données décisionnelles (Dimensional Modeling (Lien122)). Si ces tables sont uniquement reconstruites (par exemple chaque nuit) et ne font l'objet d'aucune mise à jour de type INSERT, UPDATE, DELETE entre deux chargements, pourquoi pas... En tout cas, il y aura du null, de la neige, de la redondance difficilement contrôlable malgré le soin extrême que l'on apportera aux opérations (*errare humanum est...*) Quoi qu'il en soit, les bases de données décisionnelles ne sont pas l'objet de cet article.

2.7. Dénormalisation vs amélioration (optimisation)

Le terme *dénormalisation* est bien souvent dévoyé. Chris Date met les points sur les i [Date 2007b] et mentionne quelques techniques d'amélioration (*optimisation* en français) présentées à tort comme relevant de la dénormalisation. (Dans ce qui suit, on parlera de tables plutôt que de relvars.)

1er exemple (comparable au 1er exemple du paragraphe 2.8).

Supposons que l'on ait à structurer une table des ventes journalières des magasins de l'entreprise Tartempion sur une période d'une semaine. Il est d'usage de structurer cette table, appelons-la Magasin_V, à l'aide des attributs suivants : MagId, Jour, ChiffreAffaires, le couple {MagId, Jour} étant clé. Pour améliorer les performances (principe du « tout en une seule ligne »), certains préfèrent mettre en œuvre une table, appelons-la Magasin_H, de clé {MagId}, telle que le chiffre d'affaires fasse l'objet d'un attribut pour chaque jour de la semaine. En général, le terme employé pour ce changement de structure est celui de dénormalisation, et c'est à tort, car les deux tables respectent la BCNF (et même la 5NF), elles sont bien normalisées.

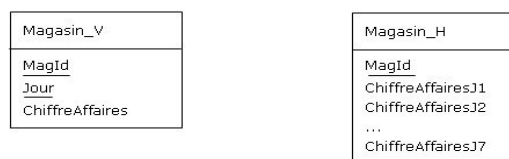


Figure 2.7 - Représentation verticale / horizontale

Maintenant, on peut faire observer que la structure de la table Magasin_H a le grave défaut de ne pas être évolutive : si les besoins de l'entreprise deviennent décennaires, quelles seront les conséquences ? Dans le cas

la table Magasin_V, ceci sera transparent, par contre il faudra changer la structure de la table Magasin_H, opération lourde s'il en est (sans parler des requêtes existantes qui devront être modifiées pour prendre en compte les attributs supplémentaires).

Du point de vue de la manipulation des données, on observera que dans le cas de la table Magasin_H, l'utilisation des opérateurs classiques d'agrégation, SUM, AVG, etc. est pour le moins remise en question.

Du point de vue de la performance, dans les deux cas, une seule lecture d'un enregistrement physique suffira pour connaître le chiffre d'affaires hebdomadaire d'un magasin. Mais, dans le cas de la table Magasin_H, il serait déraisonnable d'indexer les sept colonnes si le besoin s'en faisait sentir, en effet la performance des mises à jour est (au moins) inversement proportionnelle au nombre d'index.

En fait, l'en-tête de la table Magasin_H ressemble plutôt à celui d'un résultat à présenter à l'utilisateur. On doit conserver la structure de la table Magasin_V, et créer une **vue** VH afin de présenter les données selon la structure de la table Magasin_H, ça n'est quand même pas bien sorcier. Se reporter chez DVP à l'exemple des clients et des options ([Lien123](#)). Le mieux est encore d'en passer par un **instantané** (*snapshot*) rafraîchi à chaque mise à jour (la vue « matérialisée » (oxymore...) de SQL).

2e exemple.

Supposons que la table Magasin_V ci-dessus corresponde à un regroupement de tables par régions : Ces tables sont normalisées, elles respectent la 5NF. Pour connaître le chiffre d'affaires national, c'est l'opérateur **UNION** que l'on utilisera. On peut du reste créer une **vue** d'union ou un instantané pour définir la table virtuelle Magasin_V (puis la vue VH).

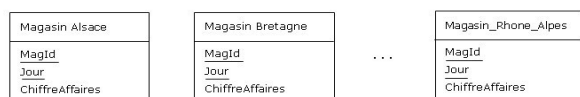


Figure 2.8 - Décomposition verticale

A noter que, si le SGBD permet le partitionnement des tables, on pourra se contenter de n'avoir qu'une table des magasins, en affectant une partition à chacun d'eux (tout en définissant au besoin une vue par magasin).

3e exemple.

Si les magasins sont en relation avec les produits, leur chiffre d'affaires par produit peut être géré de façon redondante : le chiffre d'affaires total par produit, est égal à la somme des chiffres d'affaires par magasin et par produit, il y a donc **redondance** (avec tous les risques d'incohérence inhérents). Mais ces deux tables respectent encore la BCNF (et la 5NF).

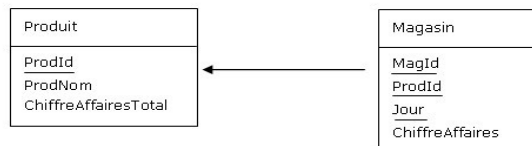


Figure 2.9 - Redondance inter-tables

N.B. Pour éviter de prendre des risques, on peut sous-traiter au SGBD le contrôle de la redondance, en attachant à la table Magasin un trigger chargé du calcul du chiffre d'affaires total des magasins concernés et qui mette ainsi à jour la table Produit en temps réel. Pour ne pas polluer cette table par les mises à jour, on peut préférer déplacer l'attribut ChiffreAffairesTotal dans un instantané, comme dans les exemples précédents.

4e exemple.

Un client passe commande. Une commande se décline en lignes de commande. Pour chaque ligne de commande, on s'engage en fonction des disponibilités des produits en stock, en cours de fabrication, des approvisionnements en cours, etc. ; un engagement est composé à son tour de parties livrables en fonction du nombre de camions nécessaires pour l'acheminement, etc.

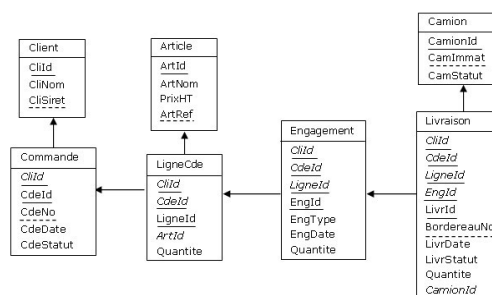


Figure 2.10 - Identification relative et redondance intra-clés

Par référence aux RVA (cf. paragraphe 2.6), les commandes représentent une propriété multivaluée du client, les lignes de commande représentent à leur tour une propriété multivaluée de la commande, les engagements sur ligne de commande une propriété multivaluée de la ligne de commande, etc. Dans le diagramme ci-dessus, on a déplié les RVA et utilisé l'identification relative (cf. note qui suit), source de redondances au sein des clés.

(N.B. Dans le diagramme, les clés primaires sont soulignées, les clés étrangères sont en italiques, sachant que les attributs appartenant à une clé primaire peuvent aussi appartenir à une clé étrangère ; les principales clés alternatives — celles qui sont connues de l'utilisateur — sont soulignées en traits discontinus).

Note concernant l'identification relative

Revenons sur l'exemple précédent. Utiliser l'identification relative revient à considérer (au niveau logique) que la clé d'une table - par exemple Commande - est composée des attributs composant la clé de la table dont elle est une propriété multivaluée (Commande est une propriété multivaluée de Client et conceptuellement parlant, il s'agit

d'une entité-type *faible*), plus un attribut permettant de distinguer chaque commande d'un client donné. Selon l'usage, cet attribut supplémentaire est de type Entier et numérote chaque commande relativement à un client. Exemple (clés soulignées) :

Client { <u>CliId</u> , CliNom, CliSiret, ...}			
1	Dubicobit	12345678900001	
2	Frichmoutz	31415926500009	
...

Commande { <u>CliId</u> , <u>CdeId</u> , CdeNo, CdeDate, ...}				
1	1	123456	15/02/2009	
1	2	234567	01/04/2009	
1	3	234575	02/04/2009	
2	1	123023	20/01/2009	
2	2	230239	17/03/2009	
2	3	256789	06/01/2010	
...

Du point de vue de l'utilisateur, le client Dubicobit a passé les commandes 123456, 234567 et 234575. Du point de vue du système, les commandes <1, 1>, <1, 2> et <1, 3> sont celles du client 1, tandis que les commandes <2, 1>, <2, 2> et <2, 3> sont celles du client 2.

Par contraste, utiliser l'**identification absolue** pour la table Commande (qui conceptuellement parlant n'est plus une entité-type faible) consiste à changer la composition de la clé, en remplaçant le couple {CliId, CdeId} par un singleton {CdeId} :

Commande { <u>CdeId</u> , <u>CdeId</u> , CdeNo, CdeDate, ...}				
1	1	123456	15/02/2009	
1	2	234567	01/04/2009	
1	3	234575	02/04/2009	
2	4	123023	20/01/2009	
2	5	230239	17/03/2009	
2	6	256789	06/01/2010	
...

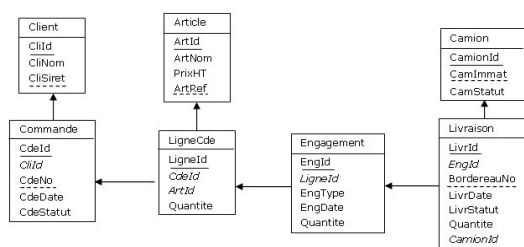


Figure 2.11 - Identification absolue systématique

L'intérêt de l'identification relative ne saute pas aux yeux, et l'on pourrait légitimement douter de sa pertinence, donc préférer ne pas la mettre en œuvre. Si les tables Commande, LigneCde, Engagement et Livraison respectent la cinquième forme normale avec l'identification relative, celle-ci est cause de **redondance** au sein des clés, ce qui n'a pas lieu quand on utilise l'identification absolue. Toutefois, cette redondance est parfaitement contrôlée par le système grâce aux contraintes d'intégrité référentielle déclarées. Par ailleurs, d'aucuns objectent que cette redondance est nécessairement consommatrice de ressources (mémoire, disque...), tandis que l'utilisation de l'identification absolue

entraînerait une consommation minimale. On peut montrer qu'il n'en est rien (cf. Annexe F.1) et que la durée de certaines requêtes peut être réduite de façon très sensible, voire déterminante si l'on passe à l'identification relative (cf. Annexe F.2).

Conséquence de l'identification relative sur l'organisation des requêtes SQL

Supposons que l'on ait besoin de savoir quels camions sont concernés par les livraisons chez le client Gillou (Siret = 12345678900001). Si on utilise l'identification absolue, on devra coder une requête SQL faisant intervenir toutes les tables intermédiaires, à savoir Commande, LigneCde, Engagement et Livraison :

Requête 1 (identification absolue)

```

SELECT DISTINCT Camion.CamImmat
FROM Client JOIN Commande
      ON Client.CliId = Commande.CliId
JOIN LigneCde
      ON Commande.CdeId = LigneCde.CdeId
JOIN Engagement
      ON LigneCde.LigneId = Engagement.LigneId
JOIN Livraison
      ON Engagement.EngId = Livraison.EngId
JOIN Camion
      ON Livraison.CamionId = Camion.CamionId
WHERE CliSiret = '12345678900001' ;
  
```

(Incidentement, comme on s'intéresse en particulier au client Gillou, l'opération est performante, mais si l'on effectue des traitements de type batch, sans que les clés étrangères fassent l'objet d'index clusters (cf. Annexes F.2 et F.3), la dégradation des performances peut poser de très gros problèmes pour la Production informatique.)

Dans le cas de l'identification relative, on pourrait aussi écrire une requête analogue :

Requête 2 (identification relative)

```

SELECT DISTINCT Camion.CamImmat
FROM Client JOIN Commande
      ON Client.CliId = Commande.CliId
JOIN LigneCde
      ON Commande.CliId = LigneCde.CliId
AND Commande.CdeId = LigneCde.CdeId
JOIN Engagement
      ON LigneCde.CliId = Engagement.CliId
AND LigneCde.CdeId = Engagement.CdeId
AND LigneCde.LigneId = Engagement.LigneId
JOIN Livraison
      ON Engagement.CliId = Livraison.CliId
AND Engagement.CdeId = Livraison.CdeId
AND Engagement.LigneId = Livraison.LigneId
AND Engagement.EngId = Livraison.EngId
JOIN Camion
      ON Livraison.CamionId = Camion.CamionId
WHERE CliSiret = '12345678900001' ;
  
```

Mais on peut alléger la requête ainsi :

Requête 3 (identification relative, variante)

```

SELECT DISTINCT Camion.CamImmat
FROM Client JOIN Commande
      ON Client.CliId = Commande.CliId
JOIN LigneCde
      ON Commande.CliId = LigneCde.CliId
JOIN Engagement
      ON LigneCde.CliId = Engagement.CliId
JOIN Livraison
      ON Engagement.CliId = Livraison.CliId
JOIN Camion
      ON Livraison.CamionId = Camion.CamionId
WHERE CliSiret = '12345678900001' ;

```

Et plus important, on améliorera plus que substantiellement la performance en prenant un raccourci façon couloir spatio-temporel, en codant encore plus simplement :

Requête 4 (identification relative, 2e variante)

```

SELECT DISTINCT Camion.CamImmat
FROM Client JOIN Livraison
      ON Client.CliId = Livraison.CliId
JOIN Camion
      ON Livraison.CamionId = Camion.CamionId
WHERE CliSiret = '12345678900001' ;

```

Dans ce genre d'exercice et dans un contexte batch ou de requêtes lourdes, l'identification absolue ne pourra évidemment pas rivaliser en performances et fera que l'on aura l'impression de faire du surplace (cf. Annexe F.3)...

Au passage, faisons observer que, grâce à la propagation de l'attribut CliId par identification relative, de lui-même l'optimiseur de DB2 for z/OS aménage les requêtes dans lesquelles les tables intermédiaires n'interviennent que comme courroies de transmission et les réécrit, les « optimise » pour produire la 4e requête.

5e exemple.

Certains auteurs merisiens sont certes des références reconnues dans leur partie, mais ils feraient mieux de garder le silence quand ils s'aventurent dans les terres relationnelles (Relationland), car ils sont manifestement mal équipés pour cela. Dans [RoMo 1989], au paragraphe 6.2.6 « Optimisation du MLD relationnel », page 200, est présenté un MCD partiel d'un système de facturation, comprenant les trois entités-types suivantes : CLIENT, FACTURE, REGLEMENT liées par des relations fonctionnelles, à savoir des CIF (contraintes d'intégrité fonctionnelle) :

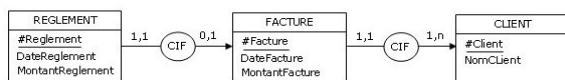


Figure 2.12 - CIF (REGLEMENT - FACTURE)

Lors du passage au MLD, est produit un ensemble de tables, dont l'équivalent reformulé en SQL peut être le suivant (on y remplace le symbole # par Id) :

```

CREATE TABLE CLIENT
(
    IdClient          Int          NOT NULL

```

```

, NomClient         VarChar(32)  NOT NULL
, CONSTRAINT CLIENT_PK PRIMARY KEY (IdClient)
) ;
CREATE TABLE FACTURE
(
    IdFacture        Int          NOT NULL
, DateFacture       Date         NOT NULL
, MontantFacture    Int          NOT NULL
, IdClient           Int          NOT NULL
, CONSTRAINT FACTURE_PK PRIMARY KEY (IdFacture)
, CONSTRAINT FACTURE_FK FOREIGN KEY (IdClient) REFERENCES CLIENT (IdClient)
) ;
CREATE TABLE REGLEMENT
(
    IdReglement      Int          NOT NULL
, DateReglement     Date         NOT NULL
, MontantReglement Int          NOT NULL
, IdFacture          Int          NOT NULL
, CONSTRAINT REGLEMENT_PK PRIMARY KEY (IdReglement)
, CONSTRAINT REGLEMENT_FK FOREIGN KEY (IdFacture) REFERENCES FACTURE (IdFacture)
) ;

```

Jusque-là tout va bien. Maintenant je cite (en notant que l'auteur remplace « CIF » par « DF », mais peu importe) :

« [...] La DF de REGLEMENT vers FACTURE a pour cardinalités maximales 1 dans chaque sens. Il est indispensable de procéder à une optimisation [...]. L'optimisation présente se traduira par l'introduction dans FACTURE de la clé étrangère #Reglement, afin de faciliter la communication entre FACTURE et REGLEMENT. »

Selon l'auteur, la structure de FACTURE doit donc **impérativement** être « optimisée » ainsi :

```

CREATE TABLE FACTURE
(
    IdFacture        Int          NOT NULL
, DateFacture       Date         NOT NULL
, MontantFacture    Int          NOT NULL
, IdClient           Int          NOT NULL
, IdReglement       Int
/* NULL imposé ! */
, CONSTRAINT FACTURE_PK PRIMARY KEY (IdFacture)
, CONSTRAINT FACTURE_FK1 FOREIGN KEY (IdClient)
REFERENCES CLIENT (IdClient)
, CONSTRAINT FACTURE_FK2 FOREIGN KEY (IdReglement)
/* « Optimisation » au
moyen d'un cycle, blurps ! */
REFERENCES REGLEMENT (IdReglement)
) ;

```

Quelle horreur ! Je ne sais pas trop ce que l'auteur entend par « faciliter la communication », en tout cas on se retrouve maintenant avec un cycle entre FACTURE et REGLEMENT, outre que l'attribut IdReglement de FACTURE doit être marqué NULL lors de la création de chaque facture, puis mis à jour avec la valeur qui va bien à chaque règlement effectif. Cette proposition calamiteuse d'« optimisation » est évidemment bonne pour la poubelle.

Ce qu'il faut faire :

1. Ne pas toucher à la structure de la table FACTURE, mais définir une clé alternative pour la table REGLEMENT :

```
CREATE TABLE REGLEMENT
(
    IdReglement      Int          NOT NULL
    , DateReglement  Date         NOT NULL
    , MontantReglement Int        NOT NULL
    , IdFacture       Int          NOT NULL
    , CONSTRAINT REGLEMENT_PK PRIMARY KEY
      (IdReglement)
    , CONSTRAINT REGLEMENT_AK UNIQUE (IdFacture)
/* Clé alternative */
    , CONSTRAINT REGLEMENT_FK FOREIGN KEY
      (IdFacture)
REFERENCES
FACTURE (IdFacture)
) ;
```

N.B. Si *nihil obstat*, au niveau conceptuel identifier REGLEMENT relativement à FACTURE, de telle sorte qu'au niveau logique l'attribut IdReglement disparaisse de REGLEMENT et que {IdFacture} y soit à la fois clé primaire et étrangère.

2. Définir au besoin une vue, appelons-la FACT_RGLT, pour effectivement « faciliter la communication », c'est-à-dire tout savoir sur les factures réglées.

Dans le style SQL, cette vue (parmi d'autres, selon les besoins) pourrait être :

```
CREATE VIEW FACT_RGLT
(IdFacture, DateFacture, MontantFacture,
IdClient,
IdReglement, DateReglement,
MontantReglement)
AS
SELECT x.IdFacture, x.DateFacture,
x.MontantFacture, x.IdClient,
y.IdReglement, y.DateReglement,
y.MontantReglement
FROM FACTURE AS x JOIN REGLEMENT AS y
ON x.IdFacture = y.IdFacture ;
```

Pour des raisons de symétrie, définir aussi une vue pour tout savoir sur les factures non réglées :

```
CREATE VIEW FACT_NON_RGLT
(IdFacture, DateFacture, MontantFacture,
IdClient)
AS
SELECT x.IdFacture, x.DateFacture,
x.MontantFacture, x.IdClient
FROM FACTURE AS x
WHERE NOT EXISTS
(SELECT ' '
FROM REGLEMENT AS y
WHERE x.IdFacture = y.IdFacture) ;
```

Et tant qu'à faire, pour disposer d'un jeu complet, définir une vue supplémentaire qui permette de présenter toutes les factures, qu'elles soient notées réglées ou non réglées :

```
CREATE VIEW FACT_STATUT
(IdFacture, DateFacture, MontantFacture,
```

```
IdClient,
IdReglement, DateReglement,
MontantReglement, Statut)
AS
SELECT IdFacture, DateFacture,
MontantFacture, IdClient,
IdReglement, DateReglement,
MontantReglement, 'Réglé'
FROM FACT_RGLT
UNION
SELECT IdFacture, DateFacture,
MontantFacture, IdClient,
' ', ' ', ' ', 'Non Réglé'
FROM FACT_NON_RGLT ;
```

Mais revenons à l'étude de la normalisation, qui est quand même le sujet de cet article.

3. Première forme normale

Beaucoup de choses vraies ou fausses ont été dites au sujet de la première forme normale (1NF), aussi un retour aux sources ne sera-t-il pas de trop. Rappelons une fois de plus que c'est Ted Codd qui a inventé et théorisé le concept de normalisation pour les bases de données, et il serait malvenu de chahuter sans raison profonde les définitions qu'il en a données. Les théoriciens du relationnel sont restés en phase avec ce qu'a écrit Codd, tout en fournissant à l'occasion quelques précisions. Mais le Modèle Relationnel de Données n'est pas figé pour l'éternité et avec le temps il a connu et connaîtra encore des évolutions. Ainsi, sans se départir de l'esprit insufflé par son inventeur, Date et Darwen ont-ils été amenés une vingtaine d'années plus tard à approfondir la 1NF, et ce avec une extrême rigueur.

3.1. La situation en 1969

Dans son tout premier article concernant le Modèle Relationnel de Données (cf. [Codd 1969] page 2), Codd fournissait un exemple (ici francisé) de représentation plate d'une relation de degré 4, décrivant les livraisons de pièces par des fournisseurs, à l'usage de projets, selon certaines quantités :

Livraison (Fournisseur Pièce Projet Quantité)			
1	2	5	17
1	3	5	23
2	3	7	9
2	7	5	4
4	1	1	12

Figure 3.1 - Les livraisons, représentation traditionnelle

Cette représentation tout à fait classique et « sage » n'appelle pas de commentaires particuliers.

Dans ce même article, Ted Codd avait écrit (page 3) :

« Nous avons traité de relations définies sur des domaines **simples** — domaines dont les éléments sont des valeurs **atomiques** (non décomposables). Les valeurs non atomiques peuvent être prises en considération dans le cadre relationnel. Ainsi, certains domaines peuvent avoir des relations comme éléments. Ces relations peuvent à leur tour être définies sur des domaines non simples, et ainsi de suite.

[...] L'adoption d'une perception relationnelle des données, autorise le développement d'un sous-langage universel de recherche, basé sur le **calcul des prédicats du deuxième ordre**.

[...] Le calcul des prédicats du deuxième ordre est nécessaire (plutôt que celui du premier ordre) parce que les domaines sur lesquels les relations sont définies peuvent à leur tour contenir des éléments qui sont des relations. »

Et reprenant l'exemple précédent, Codd propose une représentation sous forme de relations emboîtées :

```
P (Fournisseur, Q (Pièce, R (Projet, Quantité)))
```

Figure 3.2 - Les livraisons, avec emboîtement de relations

Cette fois-ci, la relation P est binaire (degré 2), ses domaines sont Fournisseur (simple) et Q (non simple). La relation emboîtée Q est à son tour binaire et ses domaines sont Pièce (simple) et R (non simple). Les domaines de la relation emboîtée R sont tous simples.

3.2. 1970 : Acte de naissance de la première forme normale

En 1970, dans son article de loin le plus cité et considéré comme celui dans lequel sont posées les fondations du Modèle Relationnel de Données [Codd 1970], Codd encourage à l'utilisation de domaines simples (page 381) ce qui permet d'éliminer le problème (s'il existe) du calcul des prédicats du deuxième ordre adopté l'année précédente :

« L'utilisation d'un modèle relationnel de données [...] permet de développer un sous-langage universel basé sur le calcul des prédicats. Si la collection des relations est en **forme normale**, alors un **calcul des prédicats du premier ordre** est suffisant. »

« Forme normale » étant à interpréter aujourd'hui comme « Première forme normale » (1NF). Le concepteur de bases de données relationnelles — qu'il en soit conscient ou non — applique la 1NF à la lettre parce que les domaines des attributs de ses tables sont simples. Même chose pour celui qui conçoit des MCD au sens Merise et qui applique la règle dite de vérification, selon laquelle : « Dans chaque occurrence d'individu-type ou de relation-type on ne trouve qu'une seule valeur de chaque propriété » [TRC 1989].

A noter que divers chercheurs et auteurs (par exemple [Jaeschke 1982], [Abiteboul 1984], [Korth 1988]) ont fait des propositions pour « étendre » le Modèle Relationnel et réintroduire la possibilité d'emboîter les relations les unes dans les autres. Les systèmes dans lesquels on s'affranchit de la première forme normale sont dits « Non première forme normale » (*Non First Normal Form*, NF² en abrégé). Il s'agit d'un sujet qui sort de notre périmètre et que nous ne traiterons pas, d'autant plus que Date et Darwen ont montré que l'emboîtement des relations (en tant que

valeurs) était possible sans avoir à étendre le Modèle Relationnel (cf. paragraphe 2.6).

1971 : Définition de la première forme normale

Dans son article de 1970, Codd décrit la 1re forme normale et c'est en 1971 qu'il en donne la définition (cf. [Codd 1971], page 31) :

« Une relation est en première forme normale si aucun de ses domaines ne peut contenir des éléments qui soient eux-mêmes des ensembles. Une relation qui n'est pas en première forme normale est dite non normalisée. »

Ainsi, une relation (ceci vaut pour une table SQL) est en première forme normale (1NF) si aucun de ses attributs ne peut prendre de valeur qui soit elle-même un ensemble, en l'occurrence une relation.

Selon les critères de Codd, si un SGBD acceptait l'instruction suivante, alors il autoriserait pertinemment le viol de la 1NF. En effet, l'attribut Messages est propre à contenir des relations (disons des tables dans un contexte SQL) :

```
CREATE TABLE Membre
(
  MbrId          INTEGER          NOT NULL
  , Pseudonyme   VARCHAR(16)     NOT NULL
  , DateInscription DATE         NOT NULL
  , Localisation  VARCHAR(48)    NOT NULL
  , AdrCourriel  VARCHAR(48)    NOT NULL
  , Messages (   MessageId      INTEGER          NOT NULL
                , MessageDate   TIMESTAMP       NOT NULL
                , MessageTexte   VARCHAR(MAX)    NOT NULL
                , Forum          CHAR(8)         NOT NULL
                )
  , CONSTRAINT MbrPk PRIMARY KEY (MbrId)
  , CONSTRAINT CK2 UNIQUE (Pseudonyme)
  , CONSTRAINT CK3 UNIQUE (AdrCourriel) ;
```

Figure 3.3 - Viol de la 1NF

Pour rester en accord avec Codd, la structure précédente peut être transformée exactement comme il le montre dans son article de 1970, en procédant par scissiparité, ce qui donne lieu à deux structures, Membre et Message :

```
CREATE TABLE Membre
(
  MbrId          INTEGER          NOT NULL
  , Pseudonyme   VARCHAR(16)     NOT NULL
  , DateInscription DATE         NOT NULL
  , Localisation  VARCHAR(48)    NOT NULL
  , AdrCourriel  VARCHAR(48)    NOT NULL
  , CONSTRAINT MbrPk PRIMARY KEY (MbrId)
  , CONSTRAINT CK2 UNIQUE (Pseudonyme)
  , CONSTRAINT CK3 UNIQUE (AdrCourriel) ;

CREATE TABLE Message
(
  MbrId          INTEGER          NOT NULL
  , MessageId     INTEGER          NOT NULL
  , MessageDate   TIMESTAMP       NOT NULL
  , MessageTexte  VARCHAR(MAX)    NOT NULL
  , Forum        CHAR(8)         NOT NULL
  , CONSTRAINT MessPk PRIMARY KEY (MbrId, MessageId)
  , CONSTRAINT MessFk FOREIGN KEY (MbrId)
    References Membre (MbrId) On Delete Cascade) ;
```

Figure 3.4 - Respect de la 1NF

(La contrainte MessFk n'est présente que pour des raisons de cohérence entre les deux tables).

Retrouvez la suite de l'article de François de Sainte Marie en ligne : [Lien124](#)

Liens

- Lien1 : <http://jcp.org/en/jsr/detail?id=334>
- Lien2 : <http://jcp.org/en/jsr/detail?id=335>
- Lien3 : <http://jcp.org/en/jsr/detail?id=336>
- Lien4 : <http://jcp.org/en/jsr/detail?id=337>
- Lien5 : <http://jcp.org/en/jsr/detail?id=203>
- Lien6 : <http://blog.developpez.com/adiguba/p8061/java/7-dolphin/nio2-file-system-api/>
- Lien7 : <http://jcp.org/en/jsr/detail?id=292>
- Lien8 : <http://blog.developpez.com/adiguba/p8415/java/7-dolphin/multicatch-rethrow/>
- Lien9 : <http://blog.developpez.com/adiguba/p9231/java/try-with-resources/>
- Lien10 : <http://blog.developpez.com/adiguba/p8012/java/7-dolphin/modif-du-project-coin/>
- Lien11 : <http://jcp.org/en/jsr/detail?id=308>
- Lien12 : <http://jcp.org/en/jsr/detail?id=310>
- Lien13 : <http://blog.developpez.com/adiguba/p8947/java/7-dolphin/public-defenders-methods/>
- Lien14 : <http://blog.developpez.com/adiguba/p9232/java/java7-projet-lambda/>
- Lien15 : <http://jcp.org/en/jsr/detail?id=260>
- Lien16 : <http://jcp.org/en/jsr/detail?id=295>
- Lien17 : <http://jcp.org/en/jsr/detail?id=296>
- Lien18 : <http://blog.developpez.com/adiguba/p9504/java/7-dolphin/jsr-java-7-java-8/>
- Lien19 : <http://mickael-lt.developpez.com/tutoriels/android/personnaliser-listview/>
- Lien20 : http://mickael-lt.developpez.com/tutoriels/android/creation-galerie-connectee/fichiers/DVP_Gallery.zip
- Lien21 : <http://android-developers.blogspot.com/2010/07/multithreading-for-performance.html>
- Lien22 : <http://mickael-lt.developpez.com/tutoriels/android/creation-galerie-connectee/>
- Lien23 : <http://www.tutomobile.fr/faire-un-navigateur-web-tutoriel-android-n%C2%B09/11/07/2010/>
- Lien24 : <http://a-renouard.developpez.com/tutoriels/android/realiser-navigateur-web/>
- Lien25 : http://www.zenika.com/conference/java/concurrent_programming_with_spring_and_david_syer?fg=50011
- Lien26 : <http://www.developpez.net/forums/d1000628/java/general-java/spring/conference-sugfr-concurrent-programming-and-distributed-applications-with-spring/>
- Lien27 : <http://www.php.net/manual/fr/mysqli.overview.php>
- Lien28 : <http://www.phpteam.net/index.php/articles/les-nouveautes-de-php-6>
- Lien29 : <http://www.php.net/manual/fr/mysqli.installation.php>
- Lien30 : <http://ca2.php.net/manual/fr/pdo.drivers.php>
- Lien31 : <http://php.net/manual/fr/pdo.getavailabledrivers.php>
- Lien32 : <http://ca2.php.net/manual/fr/function.phpinfo.php>
- Lien33 : http://www.owasp.org/index.php/Top_10_2010-Injection
- Lien34 : <http://bugs.php.net/bug.php?id=47224>
- Lien35 : <http://php.developpez.com/faq/?page=pdo#pdo-connect>
- Lien36 : <http://fmaz.developpez.com/tutoriels/php/comprendre-pdo/>
- Lien37 : <https://www.google.com/recaptcha/admin/create>
- Lien38 : <http://khayyam.developpez.com/articles/web/zend-framework/captchas/>
- Lien39 : <http://debray-jerome.developpez.com/demos/geolocalisation.html#coords>
- Lien40 : <http://debray-jerome.developpez.com/demos/geolocalisation.html#map>
- Lien41 : <http://debray-jerome.developpez.com/articles/l-api-geolocalisation-en-html5/>
- Lien42 : <http://debray-jerome.developpez.com/demos/storage.html#demo1>
- Lien43 : <http://debray-jerome.developpez.com/demos/storage.html#demo2>
- Lien44 : <http://debray-jerome.developpez.com/articles/comprendre-le-storage-en-html5/>
- Lien45 : <http://plambert.developpez.com/tutoriel/css/zoomer-image-vignette/fichiers/resultat1.html>
- Lien46 : <http://plambert.developpez.com/tutoriel/css/zoomer-image-vignette/fichiers/resultat2.html>
- Lien47 : <http://plambert.developpez.com/tutoriel/css/zoomer-image-vignette/>
- Lien48 : <http://jboss.org/jbossas/downloads/>
- Lien49 : <http://httpd.apache.org/download.cgi>
- Lien50 : <http://tomcat.apache.org/download-connectors.cgi>
- Lien51 : <http://npnoel-perez.developpez.com/tutoriel/jboss/loadbalancing/>
- Lien52 : <http://www.actionscript-facile.com/pourquoi-creer-des-composants-graphiques/article1233.html>
- Lien53 : http://matthieu-deloison.developpez.com/tutoriels/flash/actionscript-facile-ui-composants-framework-as3/composants_graphiques_chapitre1/
- Lien54 : <http://wtffs.com/>
- Lien55 : <http://www.yuiblog.com/blog/2010/02/03/video-crockonjs-1>
- Lien56 : <http://jsfiddle.net/braincracking/P7a5g/>
- Lien57 : <http://jsfiddle.net/braincracking/q6gwx/>
- Lien58 : <http://blogs.msdn.com/kristoffer/archive/2007/02/13/javascript-prototype-versus-closure-execution-speed.aspx>
- Lien59 : <http://jsfiddle.net/braincracking/6MyIV/>
- Lien60 : http://developer.yahoo.com/yui/examples/yahoo/yahoo_extend.html
- Lien61 : [http://jquery.developpeur-web2.com/documentation/fonctions-diverses/\\$.extend.php](http://jquery.developpeur-web2.com/documentation/fonctions-diverses/$.extend.php)
- Lien62 : <http://jpvincent.developpez.com/tutoriels/javascript/javascript-orientee-objet-syntaxe-base-classes-js-intention-developpeurs-php/>
- Lien63 : <http://www.admixweb.com/2009/10/25/how-to-easily-create-a-javascript-framework-part-4/>
- Lien64 : <http://kalyparker.developpez.com/articles/js/VOZ-partie-3/>
- Lien65 : http://kalyparker.developpez.com/articles/js/VOZ-partie-4/fichiers/vozpart4_fr.html
- Lien66 : http://kalyparker.developpez.com/articles/js/VOZ-partie-4/fichiers/vozpart4_fr.js
- Lien67 : <http://www.admixweb.com/>
- Lien68 : <http://kalyparker.developpez.com/articles/js/VOZ-partie-4/>
- Lien69 : http://www.boost.org/users/download/version_1_45_0
- Lien70 : <http://cpp.developpez.com/livres/?page=tous#L0321113586>
- Lien71 : <http://cpp.developpez.com/livres/?page=livresConcept#L0321334876>
- Lien72 : <http://www.research.att.com/%7EBs/abstraction.pdf>
- Lien73 : <http://cpp.developpez.com/faq/cpp/?page=boost>
- Lien74 : <http://cpp.developpez.com/cours/?page=bibliotheques#tutoriels-boost>
- Lien75 : <http://cpp.developpez.com/livres/?page=livresConcept#L0321227255>

Lien76 : <http://www.developpez.net/forums/d1006685/c-cpp/cpp/bibliotheques/boost/sortie-boost-1-45-a/>
Lien77 : <http://labs.qt.nokia.com/2010/11/09/qt-creator-2-1-beta-2/>
Lien78 : <http://qt.nokia.com/developer/qt-qtcreator-prerelease#download>
Lien79 : <http://www.developpez.net/forums/d947087/c-cpp/bibliotheques/qt/ouils/edi/traduction-francais-qt-creator-2-1-beta-2-a/>
Lien80 : <http://labs.qt.nokia.com/2010/10/26/qt-is-going-modular/>
Lien81 : <http://www.developpez.net/forums/d992306/c-cpp/bibliotheques/qt/qt-modularisation-framework-serait-larchitecture-qt-4-8-a/>
Lien82 : <http://labs.qt.nokia.com/2010/09/08/state-of-html5-canvas-in-qtwebkit/>
Lien83 : <http://philip.html5.org/tests/canvas/suite/tests/>
Lien84 : <http://www.whatwg.org/>
Lien85 : <http://www.mrspeaker.net/dev/parcycle/>
Lien86 : <http://www.bel.fi/%7Ealankila/plasma.html>
Lien87 : <http://deanm.github.com/pre3d/monster.html>
Lien88 : <http://hakim.se/experiments/html5/trail/03/>
Lien89 : <http://www.effectgames.com/demos/canvascycle/>
Lien90 : <http://www.canvasdemos.com/>
Lien91 : <http://qt-labs.developpez.com/webkit/etat-html5-canvas/>
Lien92 : <http://qt.developpez.com/evenement/2010-devdays/reportage/>
Lien93 : <http://dotnet.developpez.com/cours/?page=csharp#silverlightc>
Lien94 : <http://www.microsoft.com/express/Downloads/#2010-Visual-Phone>
Lien95 : <http://msdn.microsoft.com/en-us/library/microsoft.phone.controls.phoneapplicationframe%28VS.92%29.aspx>
Lien96 : <http://msdn.microsoft.com/en-us/library/microsoft.phone.controls.phoneapplicationpage%28VS.92%29.aspx>
Lien97 : <ftp://ftp-developpez.com/nico-pyright/tutorial/intro-wp7/TestWindowsPhone.rar>
Lien98 : <ftp://ftp-developpez.com/nico-pyright/tutorial/intro-wp7/TestWindowsPhone.rar>
Lien99 : <http://nico-pyright.developpez.com/tutoriel/vs2010/csharp/windows-phone-seven/introduction-developpement-windows-phone-seven-wp7-silverlight/>
Lien100 : <http://msdn.microsoft.com/fr-fr/library/system.componentmodel.inotifypropertychanged.aspx>
Lien101 : <http://nico-pyright.developpez.com/tutoriel/vs2010/csharp/windows-phone-seven/listbox-isolated-storage-wp7-silverlight/>
Lien102 : <http://go.microsoft.com/?linkid=9723028>
Lien103 : <http://msdn.microsoft.com/en-us/library/microsoft.phone.shell.phoneapplicationservice.state%28VS.92%29.aspx>
Lien104 : <http://msdn.microsoft.com/en-us/library/microsoft.phone.controls.phoneapplicationpage.state%28v=VS.92%29.aspx>
Lien105 : <http://msdn.microsoft.com/en-us/library/system.windows.controls.page.onnavigatedfrom%28v=VS.92%29.aspx>
Lien106 : <http://msdn.microsoft.com/en-us/library/system.windows.controls.page.onnavigatedto%28v=VS.92%29.aspx>
Lien107 : <http://blogs.msdn.com/b/delay/archive/2010/09/08/never-do-today-what-you-can-put-off-till-tomorrow-deferredloadlistbox-and-stackpanel-help-windows-phone-7-lists-scroll-smoothly-and-consistently.aspx>
Lien108 : <ftp://ftp-developpez.com/nico-pyright/tutorial/intro-wp7/ListBoxDemo.rar>
Lien109 : <ftp://ftp-developpez.com/nico-pyright/tutorial/intro-wp7/ListBoxDemo.zip>
Lien110 : <http://pierre-chatelier.developpez.com/tutoriels/mac/macosex/introduction>
Lien111 : <http://pierre-chatelier.developpez.com/tutoriels/mac/macosex/leopard/introduction>
Lien112 : <http://pierre-chatelier.developpez.com/tutoriels/mac/objectivec/migration>
Lien113 : <http://pierre-chatelier.developpez.com/>
Lien114 : <http://mac.developpez.com/livres/>
Lien115 : <http://developer.apple.com/>
Lien116 : <http://a-renouard.developpez.com/tutoriels/ios/installation-developer-tools/>
Lien117 : <http://alltouches.com/>
Lien118 : <http://www.dunod.com/interview/developper-pour-l-iphone-et-l-ipad>
Lien119 : <http://pierre-chatelier.developpez.com/tutoriels/mac/objectivec/migration/>
Lien120 : <http://ios.developpez.com/livres/>
Lien121 : <http://portal.acm.org/citation.cfm?id=1241535>
Lien122 : http://en.wikipedia.org/wiki/Dimensional_modeling
Lien123 : <http://www.developpez.net/forums/d787289/bases-donnees/decisions-sgbd/architecture-table-client-option/#post4552843>
Lien124 : <http://fsmrel.developpez.com/basesrelationnelles/normalisation/>