Developpez

Le Mag

Édition de Août - Septembre 2011.
Numéro 35.
Magazine en ligne gratuit.

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

Sommaire

Éditorial

Ce mois-ci, la rubrique Pascal fait son grand retour dans le magazine après une trop longue absence.
Vous pouvez retrouver près de dix pages sur votre langage de programmation préféré.

Profitez-en bien !

La rédaction

Article Pascal

Création d'un système de chat en Pascal

Cet article a pour but de vous apprendre à créer un système de chat entièrement en Pascal. Il fait suite au Défi Pascal 2010 : création d'un système de chat.

par **Mick605**
Page 38

Article Java

Développer des services en Java

Présentation du développement de services en Java avec le ServiceLoader.

par **Yann D'Isanto**
Page 2

Développer des services en Java

Présentation du développement de services en Java avec le ServiceLoader.

1. Introduction

Depuis sa version 6 Java SE offre des outils pour gérer simplement des services au sein des applications Java.

L'utilisation des services permet d'apporter une modularité et un meilleur découplage à votre application. En effet, l'application utilisera les services au travers d'interfaces ce qui permet de s'abstraire de leur implémentation.

Imaginons une application nécessitant une authentification de la part de l'utilisateur. Du point de vue de l'application (couche présentation), la seule chose importante est de savoir si un couple login/password est valide, peu importe comment est effectué cette validation (couche business). L'application ne s'intéresse pas à l'implémentation du mécanisme d'authentification qui doit lui rester "invisible". Ainsi même si ce mécanisme d'authentification change cela n'impactera pas le code de l'application.

2. Déclaration d'un service

La première étape consiste à déclarer le service ce qui revient tout simplement à écrire l'interface ou la classe abstraite qui sera utilisée par l'application.

Authenticator.java
```
package org.myapp;

public interface Authenticator {

    /**
     * Renvoie true si le couple login/password spécifié est valide.
     * @param login le login
     * @param password le mot de passe
     * @return true en cas d'authentification réussie, false dans le cas contraire.
     */
    boolean authenticate(String login, char[] password);
}
```

Voici un exemple d'application qui utilise ce service :

App.main()
```
Console console = System.console();
if(console == null) {
    System.err.println("no console available");
    System.exit(1);
}
Authenticator authenticator = getAuthenticator();
if(authenticator == null) {
    System.err.println("Service d'authentification introuvable");
    System.exit(1);
}
String login = console.readLine("login:");
char[] password = console.readPassword("password:");
if(authenticator.authenticate(login,password)) {
    System.out.println("Authentification réussie, bienvenue " + login + " :-)");
} else {
    System.out.println("Login ou mot de passe invalide !");
}
```

La méthode **getAuthenticator()**, qui permet de récupérer une implémentation de l'interface Authenticator, est détaillée dans la partie suivante.

3. Récupération d'une implémentation du service

Java 6 a introduit la classe ServiceLoader (Lien 01) qui permet d'effectuer une recherche (un lookup en littérature anglaise) des implémentations disponibles pour un service donné.

App.getAuthenticator()
```
/**
 * @return la première implémentation de l'interface Authenticator trouvée.
 */
public static Authenticator getAuthenticator() {
    Authenticator authenticator = null;
    // Récupération d'un ServiceLoader pour l'interface Authenticator
    ServiceLoader<Authenticator> serviceLoader = ServiceLoader.load(Authenticator.class);
    // Iterator des implémentations trouvées par le ServiceLoader
    Iterator<Authenticator> iterator = serviceLoader.iterator();
    // On récupère la première implémentation trouvée
    if(iterator.hasNext()) {
        authenticator = iterator.next();
    }
    return authenticator;
}
```

Chose importante, ce code ne fait toujours référence qu'à l'interface et reste donc là encore totalement abstrait de l'implémentation.

Le ServiceLoader utilise un cache et un système d'initialisation à la demande. En effet, l'implémentation est instanciée lors de l'appel à la méthode *next()* de l'Iterator puis mise en cache. Ainsi, en itérant une deuxième fois, on récupère la même instance du service. Il est possible de

vider le cache (et ainsi forcer la réinstanciation du service) en appelant la méthode *reload()* ou en créant un deuxième *ServiceLoader* avec *ServiceLoader.load()*.

4. Implémentation du service

Le code de l'application est maintenant terminé, il ne reste plus qu'à implémenter et publier le service d'authentification. La seule restriction imposée est d'avoir un constructeur par défaut qui sera appelé lors du chargement. En voici un exemple simple :

StubAuthenticator.java
```
package org.myapp;

public class StubAuthenticator implements
Authenticator {

    @Override
    public boolean authenticate(String login,
char[] password) {
        return "admin".equals(login) &&
"unsecure".equals(new String(password));
    }
}
```

Afin que notre implémentation puisse être récupérée par le ServiceLoader, il faut la "publier". Pour ce faire, il suffit de créer un fichier dont le nom est celui du service (nom complet de l'interface ou de la classe abstraite) dans le répertoire ressource "*META-INF/services*". Dans ce fichier on écrit le nom complet de l'implémentation.

Par exemple, on crée un fichier *META-INF/services/org.myapp.Authenticator* dans lequel on écrit la ligne suivante :

META-INF/services/org.myapp.Authenticator
```
org.myapp.StubAuthenticator
```

Tout est maintenant en place pour exécuter l'application qui utilisera le service *Authenticator* en toute transparence. Il est ainsi possible de créer une autre implémentation (par exemple un *LdapAuthenticator* qui utiliserait un annuaire LDAP) et il suffit de placer son nom dans le fichier *META-INF/services/org.myapp.Authenticator* à la place du *StubAuthenticator* pour modifier l'implémentation de l'authentification de l'application sans toucher une seule ligne de code de celle-ci.

À noter qu'il est possible d'indiquer plusieurs implémentations dans le fichier en allant simplement à la ligne entre leurs noms. L'Iterator renvoyé par le ServiceLoader permet de toutes les récupérer. On peut aussi utiliser une boucle for étendue avec le ServiceLoader qui implémente Iterable.

```
ServiceLoader<Authenticator> serviceLoader =
ServiceLoader.load(Authenticator.class);
for(Authenticator authenticator : serviceLoader)
{
    // CODE
}
```

Il est également possible d'écrire des commentaires dans le fichier en utilisant le caractère '#' (tout ce qui suit le '#' sur la ligne est ignoré).

5. Conclusion

Bien que le ServiceLoader offre une certaine ressemblance aux frameworks d'injection de dépendances, il n'est pas vraiment fait pour répondre à cette problématique. Alors qu'un framework d'injection de dépendances, comme Spring ou Guice, s'occupe de charger une implémentation donnée et précise d'une dépendance, le ServiceLoader est conçu pour récupérer un ensemble extensible d'implémentations d'un service.

Il est donc particulièrement indiqué pour le développement de plugin/modules (on peut facilement imaginer une interface *Plugin* dont on chargerait les instances avec le ServiceLoader).

Pour résumer, le ServiceLoader est un moyen standard assez simple de facilement :

- découpler son application ;
- la rendre plus flexible/modulaire.

Retrouvez l'article de Yann D'Isanto en ligne : Lien 02

Les dernières news

Java 7 disponible en version finale

Oracle publie son environnement d'exécution et le JDK 7

Après plus de quatre ans depuis la sortie de Java 6, Oracle vient de publier la version finale de Java Runtime Environment (JRE) 7.

Cette version est la première de Java SE publiée depuis la reprise du langage par Oracle suite au rachat de SUN.

Java SE 7 apporte un support pour un bon nombre de tendances qui ont déferlé dans le monde du développement informatique depuis la publication de la dernière version. Il offre une prise en charge amélioré des langages dynamique conçus pour fonctionner sur la machine virtuelle Java comme Scala et Groovy.

Java SE 7 embarque une API permettant de simplifier l'exécution d'un programme à travers des processeurs multi-cœurs. Et plusieurs autres nouveautés importantes.

Le nouveau Runtine Java 7 peut-être utilisé par les développeurs avec les environnements de développement NetBeans ou encore IntelliJ IDEA 10.5. Oracle a annoncé qu'il publiera avant la fin de l'année une mise à jour de son EDI JDeveloper pour un support de Java 7.

Le runtime Java 7 est disponible pour les systèmes d'exploitations Linux, Solaris et Windows 32 bits et 64 bits.

Oracle a également annoncé la disponibilité de la version finale du Kit de Développement de Java 7 (JDK7),

Télécharger Java 7 sur le site d'Oracle : Lien 03

Télécharger JDK 7 sur le site d'Oracle : Lien 04

Faisons un rapide tour d'horizon des nouveautés.

Le Projet Coin (Lien 05) va apporter des nouveautés au cœur du langage.

- Strings in switch

```
case "truc":
    processTruc(s);
    break;
```

- Binary integral literals

```
int value = 0b10000000; // 128
```

- Underscores in numeric literals

```
int oneMillion = 1_000_000;
//plutôt que
int oneMillion = 1000000;
```

- Multi-catch and more precise rethrow

```
catch (IOException|SQLException ex) {
    logger.log(ex);
    throw ex;
}
```

- Improved type inference for generic instance creation (diamond)

```
Map<String, List<String>> map = new
HashMap<String, List<String>>();
//pourra s'écrire plus rapidement grâce au
diamond opérateur :
Map<String, List<String>> map = new HashMap<>();
```

- try-with-resources statement

```
BufferedReader br = new BufferedReader(new
FileReader(path));
try {
   return br.readLine();
} finally {
   br.close();
}
//pourra s'écrire :
try (BufferedReader br = new BufferedReader(new
FileReader(path)) {
   return br.readLine();
}
```

- Simplified varargs method invocation

Nio2 Le gros morceau à avaler car cela va remplacer l'antique java.io.File (qui reste cependant présent) (Lien 06) par une API beaucoup plus moderne et complète. La plupart de ces nouveautés se trouvent dans le package java.nio.File (Lien 06)

- Détection de modification de fichiers grâce à la classe WatchService : Lien 07
- Une toute nouvelle API de manipulation de fichiers : Lien 08.
- Gestion des E/S asynchrones
- Enfin une copie de fichier simple

```
FileSystem default = FileSystems.getDefault();
Path source  = default.getPath("pets/cat.txt");
Path target  =
default.getPath("nicePets/nicecat.txt");
Files.copy(source, target);
```

- Un support complet des liens physiques et symboliques (si le système de fichier les supporte).
- Une gestion propre des erreurs, via des exceptions.
- Un API complète pour l'accès aux attributs des fichiers, qui supporte les fonctionnalités de chaque système (DOS et Posix) ainsi que la gestion des utilisateurs (propriétaire et liste ACL). Le tout parfaitement extensible pour supporter d'autres systèmes de fichiers via des providers. D'ailleurs ce dernier point se concrétise en standard par l'intégration du filesystem "ZIP" qui permet de traiter un fichier ZIP comme un système de fichier standard (ou presque). Ainsi pour extraire un fichier d'un ZIP on peut faire ceci (noter l'utilisation du try-with-resource) :

```
try (FileSystem zip =
FileSystems.newFileSystem(Paths.get("fi
le.zip"), null)) {
        Path source =
zip.getPath("pets/cat.txt");
        Path target =
Paths.get("nicePets/nicecat.txt");
        Files.copy(source, target);
}
```

invokeDynamic : Une amélioration de la JVM pour les langages dynamiques (Groovy par exemple) mais qui sera utilisable directement en Java via l'API java.lang.invoke (Lien 09). CGlib et JavaAssist vont sûrement beaucoup évoluer.

Concurrency and collections updates (jsr166y) : pour améliorer vos programmes multithreadés avec la classe ForkJoinPool : Lien 10

Plus anecdotique mais quand même bien sympathique : **un nouveau look** beaucoup plus moderne pour la javadoc :

Et enfin, toute l'API est pleine de petites nouveautés. Par exemple, la nouvelle classe Objects (Lien 11) que vous pouvez découvrir avec Adiguba : Lien 12.

Commentez cette news de lunatix et Hinault Romaric en ligne : Lien 13

Construire dynamiquement ses IHM Android

Cet extrait du site Android2ee (les livres de programmation pour Android : « Android A Complete Course, From Basics to Enterprise Edition ») vous permet de comprendre comment construire dynamiquement une IHM. Il vous explique comment déclarer dynamiquement des composants graphiques, les placer dans leur layout, charger des images à partir de leur nom ou URL, générer un flot de données de tests.

1. Introduction

La construction d'Interfaces Hommes-Machines dynamique est à réserver au cas très particulier de la mise en place d'écrans qui s'adaptent aux données qu'ils doivent afficher et qui ne peuvent être connues par avance.

Un exemple typique est la construction d'IHM à partir d'un flux de données XML provenant d'un serveur web. La structure du flux est connue mais pas les données (taille des listes typiquement). Il y a bien d'autres cas où cela peut être utile de savoir construire les IHM dynamiquement. Vous trouverez certainement tout un tas de cas où une construction à la volée simplifie grandement l'implémentation d'un besoin.

Dans ce cas, il est très utile de pouvoir construire une IHM qui s'adapte parfaitement au flux reçu. C'est l'objectif de cet article que de vous donner les clefs pour réaliser une construction dynamique en toute simplicité.

2. Principe

Le principe est assez similaire à la construction d'un écran quand on fait du swing (ou d'autres langages de construction graphique).

Vous définissez votre layout, vous y placez vos composants :

Ajouter un composant

```
// Définition du Layout à construire.
LinearLayout postLayout = new LinearLayout( this );
// Définition du composant Text.
TextView txvName = new TextView( this );
txvName .setText(String. format (getString(R.string. wall_name ), post .getString( "name" )));
txvName .setTypeface(Typeface. defaultFromStyle (Typeface. BOLD ));

//Définition de la façon dont le composant va remplir le layout.
LinearLayout.LayoutParams layoutParam = new LinearLayout.LayoutParams(LinearLayout.LayoutParams. FILL_PARENT , LinearLayout.LayoutParams. WRAP_CONTENT ) ;
//Ajout du composant au layout.
postLayout .addView( txvName , layoutParam );
```

Cela devient plus subtil quand on souhaite modifier une IHM existante. Dans ce cas, il vous faut d'abord supprimer le contenu du layout à reconstruire, puis le remplir à nouveau :

Supprimer le contenu d'un layout

```
//Tout d'abord, il faut détruire tous les éléments contenus dans le layout.
LinearLayout layoutOfDynamicContent = ( LinearLayout ) findViewById(R.id. layoutOfDynamicContent );
layoutOfDynamicContent .removeAllViewsInLayout();
//Ajouter les composants à ce layout. Votre IHM va être mise à jour.
```

Vous remarquerez que l'on ne détruit pas le layout, mais son contenu. C'est d'ailleurs ce contenu qui est entièrement reconstruit.

Enfin, un élément important à garder en tête est le cycle de vie de votre activité. Si vous ne mettez pas en place la sauvegarde de vos données lors des changements d'état (au sein des méthodes onPause, onResume), votre application rechargera entièrement les données à partir du serveur. Il vous faut donc être attentif à ce point.

3. Mise en place

3.1. Structure du projet

Le projet est structuré de manière classique. Votre activité est associée à un fichier de layout qui ne décrit que les éléments qui sont statiques à votre IHM.

Pour le reste rien n'est changé.

3.2. Du code par l'exemple

Étant donné qu'il n'y a pas de concept spécifique à assimiler pour construire dynamiquement des IHM à la volée. Le mieux est d'aller directement dans le code et d'expliquer celui-ci. Ce code provient du tutoriel « DynamicGui » disponible sur Android2ee.com à la rubrique « Exemples ». L'application de cet exemple a pour objectif d'afficher la liste d'éléments dont la structure est la suivante :

Nom du champ	Type de la donnée
Titre	String
Description	String
Messages	List<Message>
FromWebPicture	Drawable

Où l'objet Message est le suivant :

Titre	String
Message	String
From	String
FromPicture	Drawable

Les contraintes sont que la taille des listes est inconnue, seule la structure des données est connue par avance. De même les images sont à récupérer soit sur le web via leur URL, soit dans le dossier ressource\drawable de l'application.

3.2.1. Les fichiers usuels de l'application

3.2.1.1. Le fichier AndroidManifest

Le manifeste Android est un fichier androidManifest.xml banal, ni l'application, ni le système ne savent que vous construisez dynamiquement votre écran.

Dans cet exemple, il définit l'application, son unique activité, la version minimale du système à utiliser et pour pouvoir aller récupérer des images via leur URL, il demande l'accès à internet.

Le fichier AndroidManifest.xml

```
<? xml version = "1.0" encoding = "utf-8" ?>
< manifest xmlns:android =
"http://schemas.android.com/apk/res/android"
package = "com.android2ee.tuto.gui"
android:versionCode = "1"
android:versionName = "1.0" >
< application android:icon = "@drawable/icon"
android:label = "@string/app_name" >
< activity android:name = ".DynamicGui"
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 = "8" />
<!-- Permission pour accéder à internet
(usuel )-->
< uses-permission android:name =
"android.permission.INTERNET" />
</ manifest >
```

3.2.2. Le fichier de layout

Ce fichier décrit les composants statiques de l'IHM. Dans cet exemple, il définit :

- le layout principal de l'activité ;
- un label affichant la chaine de caractères « Bonjour » ;
- un bouton permettant de relancer la construction dynamique de l'IHM ;
- et enfin, le layout qui contiendra la vue reconstruite dynamiquement.

Ce qui donne :

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"
android:id = "@+id/mainlayout" >
< TextView android:layout_width = "fill_parent"
android:layout_height = "wrap_content"
android:text = "@string/hello"
android:id = "@+id/txvHello" />
< Button android:text = "@string/reload"
android:id = "@+id/btnReload"
android:layout_width = "wrap_content"
android:layout_height = "wrap_content" ></ Button
>
< LinearLayout xmlns:android =
"http://schemas.android.com/apk/res/android"
android:orientation = "vertical"
android:layout_width = "fill_parent"
android:layout_height = "fill_parent"
android:id = "@+id/layoutOfDynamicContent" >
</ LinearLayout >
</ LinearLayout >
```

3.2.3. Destruction de la vue et son réaffichage

Il faut tout d'abord détruire le contenu de votre layout puis le reconstruire. La méthode à utiliser pour la destruction du contenu de votre layout est removeAllViewsInLayout(); (la méthode removeAllViews() ne marche pas). Un grand merci à Yan Verdavaine de développez.com pour m'avoir ouvert les yeux. Encore une fois, merci Monsieur.

Ensuite, il vous suffit d'ajouter vos composants à ce layout. Votre IHM sera mise à jour automatiquement.

Ce qui donne :

Destruction du contenu d'un layout

```
//Tout d'abord, il faut détruire tous les
éléments contenus dans le layout.
LinearLayout layoutOfDynamicContent =
( LinearLayout ) findViewById(R.id.
layoutOfDynamicContent );
layoutOfDynamicContent .removeAllViewsInLayout();
//Ajouter les composants à ce layout. Votre IHM
va être mise à jour.
//Cf. paragraphes suivants.
//Et c'est tout votre layout qui va être mis à
jour automatiquement.
```

3.2.4. Gestion du placement des composants au sein des layouts

L'un des points-clefs dans la construction d'IHM, dynamique ou pas, est le placement des composants au sein des layouts. En effet, outre le choix du layout, il faut arriver à positionner proprement son composant en son sein. En particulier, les définitions de son comportement en largeur et hauteur (android:layout_width et android:layout_height), de sa marge, de sa gravité sont des éléments décisifs pour la mise en place de votre écran.

Dans cet article, je n'aborderai que le cas du layout LinearLayout.

Voici, le code permettant de définir un LinearLayout et de

lui ajouter un composant en gérant les marges, la gravité, sa largeur et sa hauteur.

Paramètres de placement au sein d'un layout en Java

```
// Paramètres généraux du layout :
//Déclaration du layout et instanciation de celui-ci.
LinearLayout lilPost = new LinearLayout( this );
//Définition de son orientation.
lilPost .setOrientation(LinearLayout. VERTICAL );
//Définition de son padding (le padding est la distance en pixels, entre le bord du composant et son contenu).
lilPost .setPadding(0, 15, 0, 0);
//Définition de la couleur de fond du layout.
lilPost .setBackgroundColor(0xFF007AAD);
.
.
// Paramètres spécifiques au placement du composant webPicture au sein du layout :
//Définition du paramètre de placement du composant au sein du layout et instanciation (avec la spécification de la manière dont le composant va remplir l'espace).
LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(LinearLayout.LayoutParams. WRAP_CONTENT , LinearLayout.LayoutParams. WRAP_CONTENT );
//Définition de la gravité du composant (comment il se positionne au sein de la zone qui lui est allouée).
params . gravity =Gravity. CENTER_HORIZONTAL ;
//Définition des marges du composant (distance, en pixels, autour du composant).
params .setMargins(5, 5, 5, 5);
//Ajout du composant dans le layout avec les paramètres de positionnement définis ci-dessus.
lilPost .addView( webPicture , params );
```

Ce code est équivalent au fichier xml suivant :

Paramètres de placement au sein d'un layout en xml

```
< LinearLayout android:id = "@+id/ lilPost "
android:orientation = " vertical "
android: paddingLeft = " 5 "
android: paddingRight = " 5 "
android: paddingTop = " 5 "
android: paddingBottom = " 5 "
android :background = " # 007AAD"
>
< ImageButton
android:id = "@+id/ webPicture "
android:layout_width = " wrap_content "
android:layout_height = " wrap_content "
android: layout_marginLeft = " 5 "
android: layout_marginRight = " 5 "
android: layout_marginTop = " 5 "
android: layout_marginBottom = " 5 "
android:gravity ="center_horizontal"
>
</ ImageButton >
</ LinearLayout >
```

3.2.5. Gestion du Scroll

Dans la problématique de l'affichage d'une vue construite dynamiquement se pose souvent la question du scroll. En effet, la construction dynamique implique souvent l'incapacité à prévoir la taille que prendront nos données. Il nous faut ainsi souvent prévoir une Scroll Bar.

Sous Android le composant ScrollView ne peut posséder qu'un enfant. Cet enfant se doit donc d'être votre layout contenant les composants ajoutés dynamiquement. De même il devrait remplir l'espace en largeur et en hauteur du layout principal dans lequel il se place.

Ainsi nous obtenons le code suivant :

Gestion du Scroll

```
//Tout d'abord, il faut supprimer tous les éléments du layout principal.
//Ce layout est déjà chargé par ailleurs.
layoutOfDynamicContent .removeAllViewsInLayout();
//Ensuite il faut définir et instancier la ScrollView.
//le this correspond à l'Activité contenant ces lignes.
ScrollView scvMain = new ScrollView( this );
//On définit la couleur de fond du scroll.
scvMain .setBackgroundColor(0xFFCFDBEC);
//Et on rajoute la ScrollView au layout principal en lui demandant de remplir tout l'espace disponible.
layoutOfDynamicContent .addView( scvMain , new LinearLayout.LayoutParams(LinearLayout.LayoutParams. FILL _CONTENT , LinearLayout.LayoutParams. FILL _CONTENT ) );
//Ensuite on définit le layout qui contiendra les composants ajoutés dynamiquement.
LinearLayout lilContent = new LinearLayout( this );
//code de construction dynamique de l'IHM.
//Ajout du layout à la ScrollView en lui demandant de remplir tout l'espace disponible.
scvMain .addView( lilContent , new LinearLayout.LayoutParams(LinearLayout.LayoutParams. FILL _CONTENT , LinearLayout.LayoutParams. FILL _CONTENT ) );
```

3.2.6. Gestion des composants usuels

Le composant TextView est la classe mère de quasiment tous les composants graphiques Android. Pour cette raison, cet article se focalise sur ce composant. D'autant que cet article a pour but de décrire comment générer dynamiquement une IHM, pas de comment manipuler les composants.

Le code suivant montre comment :

- déclarer et instancier le composant ;
- lui affecter un texte, une couleur de fond et une couleur de texte ;
- définir sa police de caractères et son type face (Bold, Italic, Normal) ;
- définir le texte à afficher par défaut quand le composant n'a pas de texte à afficher ;
- positionner la sauvegarde automatique de l'état du composant associé au cycle de vie de l'activité ;
- gérer son scrolling ;
- gérer sa visibilité ;
- et enfin, l'ajouter au layout qui est censé le contenir.

Gestion du composant TextView

```
//Tout d'abord, déclarez et instanciez le composant en lui passant son Contexte.
//Ici le contexte n'est autre que l'activité dans lequel l'IHM est construite, d'où le this.
```

```
TextView txvTitle = new TextView( this );
//Affectation du texte à afficher par le composant.
txvTitle .setText( "titre : " + post .getString( "titre" ));
//Affectation de sa couleur de fond.
txvTitle .setBackgroundColor(0xFF627AAD);
//Affectation de sa couleur de text.
txvTitle .setTextColor(0xFFFFFFFF);
//Définition du typeFace du composant (Italic, Bold, Normal,Bold_Italic), un autre type de typeFace étant la police : MonoSpace, Sans_Serif, Serif.
txvTitle .setTypeface(Typeface. defaultFromStyle (Typeface. BOLD ));
//Le texte à afficher quand il n'y a aucun texte à afficher par le composant (i.e. quand le composant n'a rien à afficher).
txvTitle .setHint( "C'est quoi un HintText" );
//La couleur du hint-texte.
txvTitle .setHintTextColor(0xFF555555);
//Pour permettre de sauver en mémoire l'état du composant quand l'activité sera détruite pour être restauré plus tard.
txvTitle .setFreezesText( true );
//Ajout d'une scroll bar au composant.
txvTitle .setHorizontallyScrolling( true );
//Mise en place d'une ligne de transparence horizontale (à droite et à gauche) lors du scroll du composant ou setVerticalFadingEdgeEnabled pour mettre cette ligne verticale (en haut et en bas), ou les deux. Si le composant n'a pas de scrollBar, cela ne fera rien.
txvTitle .setHorizontalFadingEdgeEnabled( true );
//Définition de la hauteur/largeur de cette ligne de transparence.
txvTitle .setLines(2);
//Gérez la visibilité du composant, il peut être Visible, Invisible (caché mais l'espace pour le composant reste réservé) ou Gone (caché et l'espace du composant est libéré et utilisé par les autres composants).
txvTitle .setVisibility(View. VISIBLE );
//Et ainsi de suite, explorez l'API.
//Et ajout du composant au layout qui le contient.
lilPost .addView( txvTitle , new LinearLayout.LayoutParams(LinearLayout.LayoutParams. FILL _CONTENT , LinearLayout.LayoutParams. FILL _CONTENT ) );
```

3.2.7. Gestion des composants images

La gestion des composants de type image implique qu'il faut récupérer l'image à partir d'une chaine de caractères représentant son nom et l'affecter au composant. Ce paragraphe explique comment effectuer cette opération dans deux cas de figure :

- l'image est une ressource de votre application (dans un dossier res\drawable-**) ;
- l'image est une ressource du web (dont l'URL est connue, de type http://adresseWeb) ;

Un composant de type image est soit ImageView soit ButtonView, cela n'a pas d'importance ici, le traitement est identique.

Dans les deux cas, le code pour créer le composant est le même, seule la récupération de l'objet Drawable diffère :

Gestion du composant ImageView

```
//Définition et instanciation du composant Image.
ImageView webPicture = new ImageView( this );
//Le drawable à afficher.
Drawable monDrawable =_//Le Drawable à affecter ;
//Affectation du composant à l'image.
webPicture .setImageDrawable( //Le Drawable à affecter );
//Définition de la couleur de fond du composant (facultatif).
webPicture .setBackgroundColor(0xFFFFFFFF);
```

3.2.7.1. Cas d'une image contenue dans les ressources de l'application

Pour récupérer de manière dynamique une image contenue dans le dossier de ressources à partir de la chaine de caractères représentant son nom, il faut réussir à retrouver son identifiant. Muni de cet identifiant le chargement de la ressource est naturel.

Récupération de l'image (Drawable) à partir de son nom (String)

```
int pictureId =getResources().getIdentifier( message .getString( "fromPicture" ), "drawable" , "com.android2ee.tuto.gui" );
picture .setBackgroundDrawable(getResources().getDrawable ( pictureId ));
```

La méthode getResources appartient à la classe ContextWrapper dont étend Activity (entre autres) et renvoie un objet de type Resources. Cette classe (Resources) permet de manipuler les ressources de l'application, en particulier leur récupération.

3.2.7.1.1. La méthode getIdentifier

La méthode public int **getIdentifier** (String name, String defType, String defPackage) permet de retrouver l'identifiant généré par le compilateur pour une ressource particulière. Il suffit de lui passer le nom de la ressource, son type et le package racine de votre application (ou de l'application dans laquelle se trouve la ressource). Pour ce qui est du type, ce paramètre n'est autre que celui que l'on retrouve en seconde position dans l'appel usuel à un identifiant, par exemple :

- R.string.maChaine a pour type string ;
- R.layout.monLayout a pour type layout ;
- R.drawable.monImage a pour type drawable ;
- R.id.monComposant a pour type id.

Pour comprendre vraiment, il suffit de regarder le fichier de ressources R généré lors de la compilation :

Le fichier généré R

```
/* AUTO-GENERATED FILE. DO NOT MODIFY.
*
* This class was automatically generated by the
* aapt tool from the resource data it found. It
* should not be modified by hand.
*/

package com.android2ee.tuto.gui;
public final class R {
public static final class attr {
}
```

```
public static final class drawable {
public static final int icon =0x7f020000;
public static final int icon_sti1 =0x7f020001;
public static final int icon_sti2 =0x7f020002;
public static final int icon_sti2b =0x7f020003;
public static final int icon_sti3 =0x7f020004;
}
public static final class id {
public static final int btnReload =0x7f050002;
public static final int layoutOfDynamicContent
=0x7f050003;
public static final int mainlayout =0x7f050000;
public static final int txvHello =0x7f050001;
}
public static final class layout {
public static final int main =0x7f030000;
}
public static final class string {
public static final int app_name =0x7f040001;
public static final int hello =0x7f040000;
public static final int reload =0x7f040002;
}
}
```

La méthode **getIdentifier** (String name, String defType, String defPackage) devient alors limpide :

- le paramètre defPackage est le package du fichier ;
- le paramètre defType est le nom de classe contenant votre identifiant (la **public static final class**) ;
- le paramètre name est le nom de l'identifiant contenu dans cette classe.

3.2.7.2. Cas d'une image provenant du web

Pour récupérer une image hébergée sur le web, je préconise l'utilisation de la méthode suivante :

Récupération d'une image (Drawable) sur internet à partir de son URL

```
/**
* Cette méthode renvoie le Drawable associé à une
URL d'image
* @param urlPath l' URL de l'image
* @return L'objet Drawable téléchargé à partir de
l' URL passée en paramètre
*/
private Drawable getPicture (String urlPath ) {
// Le drawable à renvoyer
Drawable drawable = null ;
try {
// Récupération de l'URL à partir de sa
représentation sous forme de String.
URL URL = new URL ( urlPath );
//Ouverture de l'inputStream associé à cette URL
pour sa lecture.
InputStream is = (InputStream) URL .getContent();
// Construction du Drawable à partir de ce flux
entrant.
drawable = Drawable. createFromStream ( is ,
"src" );
} catch (IOException e ) {
Log. e ( tag , e .toString());
//Si une exception se produit faire quelque chose
d'intelligent.
}
// Renvoyer le résultat.
return drawable ;
}
```

Cette méthode ne fait rien de plus qu'ouvrir un flux entrant sur l'URL de l'image et à partir de ce flux construit l'objet graphique Drawable souhaité.

L'utilisation de cette méthode est elle aussi « triviale » :

Instanciation du composant ImageView

```
// Instanciation du composant image.
ImageView webPicture = new ImageView( this );
//Récupération du Drawable par appel à la méthode
getPicture et affectation au composant.
webPicture
.setImageDrawable(getPicture( http://nomDeLImage
));
```

3.2.8. Mise en place de données factices pour les tests

Je préconise, lors de la mise en place d'une IHM construite dynamiquement, l'utilisation d'une classe générant des données de tests durant le développement et les tests. Cette classe renvoie un flux aléatoire de données mais dont la structure est celle attendue. Cela permet de coder son IHM sans se soucier de la récupération de données et ainsi de se concentrer uniquement sur sa définition. Ce n'est pas nécessaire, c'est juste plus facile.

Dans l'exemple associé à cet article, la structure des données est la suivante :

Nom du champ	Type de la donnée
Titre	String
Description	String
Messages	List<Message>
FromWebPicture	Drawable

Où l'objet Message est le suivant :

Titre	String
Message	String
From	String
FromPicture	Drawable

Le Format JSON est celui choisi pour cet exemple.

La méthode de construction d'un objet de ce type est la suivante :

Construction d'un objet JSON

```
private JSONObject generateOneData () {
try {
//Instanciation et déclaration de l'objet JSON.
JSONObject jsonPost = new JSONObject();
//Récupération d'un entier de référence (le choix
du sujet du post).
int selectedInt = getRandomInt(6);
//Instanciation de ce sujet.
String subject = titres [ selectedInt ];
//Ajout du champ name avec la valeur post à
l'objet JSON.
jsonPost .put( "name" , "post" );
//Ajout du champ titre avec la valeur subject à
l'objet JSON.
```

```
jsonPost .put( "titre" , subject );
//Ajout du champ name avec la valeur post à
l'objet JSON.
jsonPost .put( "description" , descriptions
[ selectedInt ]);
//Ajout du champ FromWebPicture avec la valeur le
nom de l'URL de l'image à l'objet JSON.
jsonPost .put( "fromWebPicture" , webDrawables
[ selectedInt ]);
//Déclaration et instanciation d'un tableau JSON.
JSONArray jsonMessagesList = new JSONArray();
//Déclaration de l'objet JSON qui sera ajouté au
tableau.
JSONObject jsonMessage ;
//entier permettant d'associer la même image au
même champ « from » .
int fromNum ;
//Création en boucle des objets JSON à insérer
dans le tableau.
for ( int i = 0; i < getRandomInt(9); i ++) {
//Instanciation de l'objet JSON, ajout de valeurs
et ajout à la liste :
jsonMessage = new JSONObject();
jsonMessage .put( "name" , "message" );
jsonMessage .put( "titre" , titresMessages
[getRandomInt(6)] + " " + subject );
jsonMessage .put( "message" , messages
[getRandomInt(7)] + " " + subject );
fromNum =getRandomInt(4);
jsonMessage .put( "from" , from [ fromNum ]);
jsonMessage .put( "fromPicture" , drawables
[ fromNum ]);
jsonMessagesList .put( jsonMessage );
}
//Ajout du tableau à l'objet JSON initial.
jsonPost .put( "messages" , jsonMessagesList );
//Et retour.
return jsonPost ;
} catch (JSONException e ) {
//Faire quelque chose d'intelligent ici :
return null ;
}
}
```

Où les tableaux suivants sont utilisés comme données de test (pour des raisons de lisibilité, je les ai vidés de leur contenu) :

Tableaux des données de tests

```
/**
* 6 items Titres. Items utilisés lors de la
génération des données.
*/
String[] titres = { "" };
/**
* 6 items Descriptions. Items utilisés lors de la
génération des données.
*/
String[] descriptions = { "" };
/**
* 6 items titre du message. title Items utilisés
lors de la génération des données.
*/
String[] titresMessages = { "" };
/**
* 7 items corps du message. Items utilisés lors
de la génération des données.
*/
String[] messages = { "" };
/**
* 4 items auteur du message. Items utilisés lors
de la génération des données.
*/
String[] from = { "" };
/**
* 4 items image de l'auteur. Items utilisés lors
de la génération des données.
*/
String[] drawables = { "" };
/**
* 5 items image du sujet en provenance du web.
Items utilisés lors de la génération des données.
*/
String[] webDrawables = { "" };
```

Retrouvez l'article de Mathias Seguy en ligne : Lien 14

Créer une page de login et vérifier l'identification en communiquant avec la base de données via un script PHP

Le but de ce tutoriel est de vous apprendre à communiquer avec votre base de données en passant par un script en PHP dans votre application Android. Attention, votre script PHP devra être stocké sur un serveur.

1. Vue globale

L'activité principale appelle l'activité de Login et attend le résultat. Si cette dernière répond et les identifiants sont corrects, alors la fonction startup() sera appelée et la textView affichera les identifiants de l'utilisateur connecté.

2. C'est parti

2.1. Création de la base de données

Tout d'abord créons la base de données. Voici le script :

```
-- phpMyAdmin SQL Dump
-- version 2.6.4-pl4
-- http://www.phpmyadmin.net
--
-- Host: localhost
--

--
-- ----------------------------------------------------
-------

--
-- Création Table `auth_table`
--

CREATE TABLE IF NOT EXISTS auth_table (
  user_id int(10) UNSIGNED NOT NULL
AUTO_INCREMENT,
  username varchar(20) NOT NULL DEFAULT '',
  password varchar(32) NOT NULL DEFAULT '',
  PRIMARY KEY  (user_id),
  UNIQUE KEY username (username)
) TYPE=InnoDB ;
```

```
--
-- Insertion Table `auth_table`
--

INSERT INTO auth_table (username, password)
VALUES ('test', MD5('pass'));
```

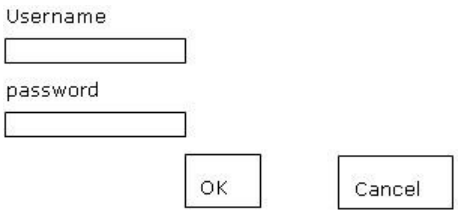
2.2. Création du script PHP

La communication avec la base de données se fait via un script en PHP situé sur votre serveur.
Vous avez donc compris que notre application va communiquer avec le script PHP afin de savoir si l'authentification est correcte.
Voici le script PHP :

```
<?php
    unset($_GET);

    if( isset($_POST['username']) &&
isset($_POST['password']) ) {

        echo '<?xml version="1.0"?>'."\n";
        echo "<login>\n";

    // host doit être remplacé par le serveur de
la base de données.
    // user représente le nom d'utilisateur de la
base de données.
    // pass est le mot de passe pour accéder à
cette base de données avec cette
        // utilisateur.
        if (!@mysql_connect('host', 'user',
'pass')) { error(1); }

        // database représente le nom d ela base
de données
            if (!mysql_select_db('database'))
{ error(2); }

        if(get_magic_quotes_gpc()) {
            $login =
stripslashes($_POST['username']);
            $pass  =
stripslashes($_POST['password']);
        } else {
            $login = $_POST['username'];
            $pass  = $_POST['password'];
        }

        unset($_POST);

        $kid = login($login, $pass);
        if($kid == -1) {
            error(3);
        } else {
            printf('    <user id="%d"/>'."\n",
$kid);
        }

        echo "</login>";
    }

    function error($ec) {
        printf('    <error
value="%d"/>'."\n".'</login>',$ec);
        die();
    }

    function login($login, $pass) {
        $select = "
                    SELECT user_id
                    FROM auth_table
                    WHERE username = '%s' AND
password = '%s'
                        ";
                $fixedlogin =
mysql_real_escape_string($login);
                $fixedpass  =
mysql_real_escape_string($pass);
                $query = sprintf($select,
$fixedlogin, $fixedpass);

        $result = mysql_query($query);
        if(mysql_num_rows($result) != 1) { return
-1; }
        $row = mysql_fetch_row($result);
        return $row[0];
    }
?>
```

2.3. Création de la vue

Ceci étant fait, créons la vue principale (main.xml).
La vue ressemblera à ceci :



Voici le code :

```
<?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"
    android:gravity="center">

  <TextView
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="Username"
  />
  <EditText
   android:id="@+id/username"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:singleLine="true"
   android:fadingEdge="horizontal"
   android:layout_marginBottom="20dip"
  />
  <TextView
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="Password"
  />
  <EditText
   android:id="@+id/password"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:password="true"
   android:singleLine="true"
```

```
   android:fadingEdge="horizontal"
  />
  <LinearLayout xmlns:android=
"http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="bottom">
    <Button
   android:id="@+id/okbutton"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="Login"
    />
    <Button
   android:id="@+id/cancelbutton"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="Cancel"
    />
  </LinearLayout>
</LinearLayout>
```

2.4. Activité principale

Le code de la page principale est plutôt simple. En effet, il appelle la page de Login et traite la réponse de cette dernière dans la fonction onActivityResult. Si l'authentification est correcte, la fonction startup() est appelée.
Voici le code :

```
package de.demo.main;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;
import de.demo.login.Login;

public class Main extends Activity
{
  private TextView tv;
    public static final int RESULT_Main = 1;

    public void onCreate(Bundle icicle)
    {
    super.onCreate(icicle);

          //Appel de la page de Login
    startActivityForResult(new Intent(Main.this,
Login.class), RESULT_Main);

    tv = new TextView(this);
    setContentView(tv);
  }

  private void startup(Intent i)
    {
          // Récupère l'identifiant
          int user = i.getIntExtra("userid",-1);

      //Affiche les identifiants de l'utilisateur
    tv.setText("UserID: "+String.valueOf(user)+"
logged in");
  }

  protected void onActivityResult(int
requestCode, int resultCode, Intent data)
    {
    if(requestCode == RESULT_Main && resultCode
== RESULT_CANCELED)
      finish();
    else
      startup(data);
  }
}
```

2.5. Activité secondaire - Login

L'activité Login est plus compliquée car il faut communiquer avec le script PHP une fois que les identifiants sont entrés et que l'utilisateur clique sur le bouton OK.
Voir le code en ligne : Lien 15

2.6. Manifest

Bon tout est fait il manque juste le manifest et vous pourrez tester votre application d'authentification !
Voici le manifest :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res
/android"
  package="de.demo.main"
  android:versionCode="1"
  android:versionName="1.0.0">
  <application android:icon="@drawable/icon"
android:label="LoginDemo">
    <activity android:name="de.demo.main.Main"
android:label="LoginDemo">
      <intent-filter>
        <action
android:name="android.intent.action.MAIN" />
        <category
android:name="android.intent.category.LAUNCHER" /
>
      </intent-filter>
    </activity>
    <activity android:name="de.demo.login.Login"
android:label="LoginDemo">
      <intent-filter>
        <action
android:name="android.intent.action.VIEW" />
        <category
android:name="android.intent.category.DEFAULT" />
      </intent-filter>
    </activity>
  </application>
<uses-permission
android:name="android.permission.INTERNET"></uses
-permission>
</manifest>
```

3. Conclusion

Et voilà, c'est terminé !
Merci d'avoir suivi ce premier tutoriel et à bientôt ;).
NB : Le code est récupéré du site anddev.org. Les commentaires y ont été ajoutés.

Retrouvez l'article de Silvera David en ligne : Lien 16

Eclipse 3.7 Indigo disponible

Support de GIT, WindowBuilder, M2Eclipse et 62 projets mis à jour

Une nouvelle version d'Eclipse est disponible. Elle porte le nom d'Eclipse Indigo.

De nombreux ajouts ont été apportés dont les plus significatifs sont certainement :

- EGIT1.0 (un client pour GIT) ;
- WindowBuilder (un outil de construction d'IHMs) ;
- M2E (le client Maven) ;
- et 62 projets qui ont été mis à jour.

La liste des nouveautés se trouve ici : What's New in 3.7 (Lien 17)

Pour accompagner cette sortie, de nombreux événements gratuits (des Eclipse DemoCamps Indigo) sont organisés un peu partout dans le monde.

En France, trois Eclipse DemoCamps Indigo se dérouleront à Grenoble (Lien 18), à Nantes (Lien 19) et à Toulouse (Lien 20).

Télécharger Eclipse Indigo sur le site de la communauté : Lien 21

Commentez cette news de Gueritarish en ligne : Lien 22

Formatage des données en PHP

Cet article a pour but de :
- donner une vision globale des formats des données en PHP ;
- aider le développeur à comprendre quel format choisir en fonction de chaque cas d'utilisation ;
- proposer des solutions pour simplifier la gestion des formats des données.

1. Définition d'un format

Ce que nous appelons "format" est en fait l'ensemble des modifications apportées à une donnée avant de commencer réellement à travailler dessus ou à l'afficher.
Par exemple, transformer les chevrons (<) en entité HTML (<) est un formatage de notre donnée.
Nous allons nous intéresser dans ce qui suit aux différents formats qu'une donnée peut avoir tout au long de l'exécution du code PHP.

2. Théorie sur le format de stockage des données

La plupart des applications PHP ont besoin d'écrire / lire / modifier / supprimer des données dans une base SQL ou dans un fichier.

Le choix du format de représentation d'une donnée est important :
si vous cryptez votre donnée, vous devrez la décrypter pour la lire ; de la même manière, si vous stockez votre donnée de manière "formatée", vous devrez la "déformater" pour la modifier.

De plus, la plupart des formatages augmentent la longueur de la donnée formatée. Donc si vous avez une base de données avec une colonne de taille X, et que vous souhaitez stocker une donnée formatée, vous ne pourrez stocker dans la base que X-F caractères, F étant le nombre de caractères nécessaires au formatage.
Il est difficile de dire à l'utilisateur de votre site Web qu'il peut saisir moins de caractères dans son formulaire s'il utilise des guillemets, des chevrons, ou d'autres métacaractères, parce que votre méthode de formatage va augmenter la taille de sa donnée.

Nous conseillons donc de toujours stocker la donnée non formatée, dans son format brut.

Il est possible de stocker des données formatées pour des raisons de performances, afin d'éviter d'avoir à effectuer le formatage à chaque affichage, mais on prendra soin de ne jamais accéder au champ formaté pour le modifier et de garder une copie de la donnée non formatée pour cela.

3. Les différents formats

Pour mettre en évidence les différents formats, nous utiliserons des exemples sur la chaîne simplifiée de test suivante : **a'a"a<a>aa\a**.
Une chaîne de test plus complète est donnée dans la dernière partie de l'article (§7.1).

Nous donnerons également les méthodes correspondantes en PHP pour formater les données lorsqu'elles existent (de manière non exhaustive).

On pourra aussi se référer à la partie "Encodage des caractères" du cours PHP de Yogui (Lien 23) sur developpez.com pour plus de détails sur les différentes fonctions présentées ici.

3.1. Le format "Raw"

Ce que nous appelons "Raw" est en fait le format qui correspond à la donnée brute, telle qu'elle a été saisie par l'utilisateur, sans aucune transformation.
Exemple : **a'a"a<a>aa\a**

3.2. Le format "Échappé" (Escape)

Il s'agit ici d'ajouter un backslash (\) devant certains métacaractères.
La méthode à appeler dépend de l'utilisation souhaitée.
Méthodes en PHP : addslashes (Lien 24), mysql_real_escape_string (Lien 25), pg_escape_string (Lien 26), escapeshellcmd (Lien 27), escapeshellarg (Lien 28), quotemeta (Lien 29), etc.
Exemple : **a\'a\"a<a>aa\\a**

3.3. Le format "HTML"

Il s'agit ici de transformer certains caractères spéciaux pour qu'ils s'affichent correctement dans une page quel que soit le jeu de caractères utilisé et qu'ils ne soient pas considérés comme des balises HTML.
Méthodes en PHP : htmlentities (Lien 30), htmlspecialchars (Lien 31).
Exemple : **a'a"a<a>aa\a**

3.4. Le format "URL"

Il s'agit ici de transformer certains caractères spéciaux pour qu'ils puissent être utilisés dans une URL.
Méthodes en PHP : urlencode (Lien 32), rawurlencode (Lien 33).
Exemple : **a'a%22a%3Ca%3Ea%3C%2Fa%3Ea%5Ca**

3.5. Le format "base64"

Il s'agit ici de transformer une chaîne en une suite de caractères simples, sans accents ou signes spéciaux, pour

qu'ils puissent être envoyés sur le réseau ou à une autre application (par exemple Flash).
Méthodes en PHP : base64_encode (Lien 34).
Exemple : **YSdhImE8YT5hPC9hPmFcYQ==**

4. Le formatage automatique - magic_quotes

La plupart du temps les données resteront dans leur format d'origine lors de leur manipulation.
Cependant, il y a une exception.
Il s'agit de trois vieilles options de PHP, donc en principe vous ne devriez plus avoir de problèmes avec :
- magic_quotes_gpc : Lien 35 ;
- magic_quotes_sybase : Lien 36 ;
- magic_quotes_runtime : Lien 37.

Elles ont toutes été marquées DEPRECATED en PHP 5.3.0 : Lien 38.
Les deux premières options, lorsqu'elles étaient activées, modifiaient automatiquement les variables lues depuis les formulaires GET et POST ainsi que les données venant de COOKIE, en appliquant la méthode "addslashes" dessus.
L'option magic_quotes_runtime modifiait quant à elle le retour de nombreuses fonctions de lecture de données, dont : file, gets, exec, mysqli_fetch_* (mais pas mysql_fetch_* !!).

PHP a fini par revenir sur ces options pour plusieurs raisons :
- Performance : toutes les données étaient échappées, alors que toutes les données n'en avaient pas besoin ;
- Lourdeur du code : lorsque le remplacement n'était pas souhaité, il fallait appeler explicitement stripslashes() à la lecture de la variable ;
- Sécurité : l'ajout des backslash () par la méthode addslashes avait pour but de protéger les requêtes des injections SQL, mais cela ne fonctionnait pas de toute façon car la méthode addslashes ne gérait pas correctement les caractères multi-octets.

De plus, dans la mesure où il s'agissait d'options de configuration, un site web pouvait très bien fonctionner correctement sur un serveur et pas du tout sur un autre avec une configuration différente.

5. La destination des données

Pour chaque cas, on donnera un exemple de donnée formatée ainsi que le risque encouru si le formatage n'est pas effectué.

5.1. L'affichage sur la page

L'affichage d'une donnée sur une page HTML doit se faire en échappant les entités HTML à l'aide de htmlentities (Lien 30) (ou htmlspecialchars : Lien 31) :
- htmlentities convertira également les accents en entité HTML, ce qui vous assurera une meilleure compatibilité de vos pages web : Lien 30 ;
- htmlspecialchars se contentera de convertir les métacaractères HTML : Lien 31.

Il est important pour des raisons de sécurité de spécifier le bon encodage à l'aide du paramètre optionnel "charset".

Si vous êtes amenés à utiliser les fonctions inverses, html_entity_decode (Lien 39) ou htmlspecialchars_decode (Lien 40), c'est que vous avez mal conçu votre application.

Donnée brute : a<u>a</u>a
Donnée formatée : a<u>a</u>a
Risque : faille de sécurité de type Cross Site Scripting (XSS) : Lien 41.

Lien vers la documentation PHP (Encodage d'une valeur d'un formulaire ou d'une URL) : Lien 42.

5.2. L'affichage dans une balise HTML

Vous devez ici protéger essentiellement les caractères de séparation utilisés dans la balise HTML, c'est-à-dire les guillemets (") ou les quotes ('), mais aussi les autres caractères spéciaux pour un affichage correct.
Vous pouvez donc utiliser htmlspecialchars (Lien 31) ou htmlentities (Lien 30).

Il est important pour des raisons de sécurité de spécifier le bon encodage à l'aide du paramètre optionnel "charset".

Si vos balises HTML utilisent des quotes (') et non des guillemets (") il faudra utiliser l'option ENT_QUOTES.

Donnée brute : a'a"a
Donnée formatée : a'a"a
Risque : faille de sécurité de type Cross Site Scripting (XSS) : Lien 41.

Lien vers la documentation PHP (Encodage d'une valeur d'un formulaire ou d'une URL) : Lien 42.

5.3. L'utilisation dans une URL

Une variable qui est utilisée pour construire une URL doit être encodée avec urlencode (Lien 32) ou rawurlencode (Lien 33).

Il est conseillé d'utiliser rawurlencode plutôt que urlencode. En effet, urlencode utilise l'ancienne norme de formatage des URL et non la norme actuelle, respectée par rawurlencode depuis PHP 5.3.0, qui est la RFC 3986 : Lien 43.
La différence la plus courante entre les deux méthodes est que urlencode va remplacer les espaces dans l'URL par le signe "+" alors que rawurlencode va les remplacer par la chaîne "%20".

Attention, seules les données doivent être encodées avec rawurlencode ! Les caractères de séparation ?, = et & ne doivent pas être encodés.
Il est possible d'encoder le nom de la page ou le nom de la variable s'ils contiennent un ou plusieurs caractères spéciaux.

Donnée brute : a"a
Donnée formatée : a%22a
Risque : faille de sécurité de type Cross Site Scripting (XSS) : Lien 41.

Lien vers la documentation PHP (Encodage d'une valeur d'un formulaire ou d'une URL) : Lien 42.

5.4. L'utilisation dans du code JavaScript

Lorsque vous avez une variable PHP qui sert à construire du code JavaScript, il est conseillé d'utiliser des guillemets pour entourer la variable PHP et d'utiliser addslashes (Lien 24).
Donnée brute : a'a"a
Donnée formatée : a\'a\"a
Risque : faille de sécurité de type Cross Site Scripting (XSS) : Lien 41.

Lien vers la documentation PHP (Passage de variable de JavaScript à PHP) : Lien 44.

5.5. L'utilisation dans une requête SQL

Pour échapper correctement une donnée destinée à être utilisée dans une requête SQL il faut absolument utiliser les méthodes fournies par la base de données utilisée :
- mysql_real_escape_string pour les bases MySQL : Lien 25;
- mysqli_real_escape_string pour les connexions mysqli : Lien 45;
- pg_escape_string pour les bases PostgreSQL : Lien 26.

Donnée brute : a'a"a
Donnée formatée : *jamais destinée à être affichée, dépend de la base de données*
Risque : faille de sécurité de type Injection SQL : Lien 46.

5.6. Le stockage dans une base de données

Comme expliqué précédemment, le stockage doit se faire au format de saisie. Cependant, pour inclure la donnée dans une requête, on doit utiliser les mêmes méthodes que pour construire une requête (§ 5.5).

Donnée brute : a'a"a
Donnée formatée : *jamais destinée à être affichée, dépend de la base de données*
Risque : faille de sécurité de type Injection SQL : Lien 46.

5.7. L'utilisation dans un mail au format HTML

Souvent oublié lors des tests, lorsqu'on génère le corps d'un mail au format HTML, le contenu ne doit pas être protégé par htmlspecialchars ou htmlentities. Ici on souhaite en général que le code HTML soit interprété.

Donnée brute : a<u>a</u>a

Attention tout de même que le contenu du mail ne vienne pas d'un utilisateur. Sinon il est plus prudent de protéger le contenu non sécurisé avec htmlspecialchars (Lien 31) ou htmlentities (Lien 30).

Risque : faille de sécurité de type Cross Site Scripting (XSS) : Lien 41.

5.8. L'utilisation dans un appel système

Peu utilisé, mais très dangereux si on l'utilise et qu'on l'oublie !

Les paramètres d'appels des méthodes suivantes doivent être formatés :
- exec : Lien 47 ;
- system : Lien 48 ;
- passthru : Lien 49 ;
- proc_open : Lien 50 ;
- shell_exec : Lien 51 ;
- ` ` (guillemets obliques, ou backquote : Alt Gr+7) : Lien 52.

Ici on doit utiliser les méthodes escapeshellarg (Lien 28) et escapeshellcmd (Lien 27).

Donnée brute : a#a$a;a
Donnée formatée : a\#a\$a\;a
Risque : faille de sécurité permettant de lancer n'importe quelle commande sur le serveur...

5.9. L'utilisation dans une expression régulière

Là encore, l'utilisation est rare.
Si vous construisez dynamiquement une expression régulière avec des données venant de l'utilisateur, il peut être intéressant de les protéger pour que votre expression fonctionne.

Pour cela, on utilise preg_quote (Lien 53) ou quotemeta (Lien 29).

Donnée brute : a.a$a*a
Donnée formatée : a\.a\$a*a
Risque : code non fonctionnel et/ou faille applicative.

5.10. Encodage pour transfert ou pièce jointe

Lorsque vous avez besoin de transférer une donnée à une autre application (Flash par exemple) ou d'ajouter une pièce jointe à un mail envoyé en PHP, vous pouvez utiliser le format base64 avec base64_encode (Lien 34). Il s'agit d'un format qui transforme complètement la chaîne en entrée mais qui assure que la chaîne encodée ne contient que des caractères simples (et donc qu'on n'aura pas de problème d'encodage, quel que soit le charset utilisé).
Pour décoder la chaîne on utilisera la méthode base64_decode (Lien 54).

Donnée brute : a'a"a<a>aa\a
Donnée formatée : YsdhImE8YT5hPC9hPmFcYQ==
Risque : code non fonctionnel et/ou faille applicative.

6. Solutions

6.1. Manuellement, au cas par cas

C'est le cas de la plupart des sites web. Le formatage de la donnée est totalement oublié et on le rajoute uniquement lorsqu'on détecte un problème (un utilisateur s'inscrit avec une apostrophe dans le login par exemple).
Inutile de dire que ce genre de site n'est pas sécurisé.

6.2. Interdire au lieu de gérer

La solution la plus simple, c'est d'interdire tous les caractères pouvant poser problème.
Maintenant, ce n'est pas forcément souhaitable d'avoir un site Web sans aucune apostrophe et autre métacaractère. Certains sites le gèrent en remplaçant les apostrophes par des underscores (_) mais ce n'est pas très propre quand même...

6.3. Tout échapper

Une autre solution "simple" :
Elle consiste à ne pas réfléchir et à filtrer toutes les variables sans exception, tout le temps, avec toutes les méthodes possibles.
Il est par contre probable que l'utilisateur voit quelque chose comme ça :
Donnée saisie : a'a"<u>a</u>a
Donnée affichée : a\\\'a\\\"a<u>a</u>a

Évidemment, ce n'est pas très joli... mais je tenais à en parler parce que j'ai déjà reçu des mails de sites parlant de sécurité informatique avec trois antislashs devant un guillemet.
Personnellement, quand je vois ça, je ne me dis pas que le site est sécurisé, mais que le webmaster manque de compétence et de confiance et ne sait pas réellement ce qu'il fait.

6.4. Cas particulier de la base de données : PDO

Il existe une autre solution qui permet d'éviter (presque) tous les problèmes d'injection SQL : passer par une abstraction de base de données comme PDO et utiliser les requêtes préparées.

Lorsque vous utilisez une requête préparée et que vous lui passez des paramètres, vous passez les paramètres et la requête séparément à la base de données. Il est donc impossible pour celle-ci de les "confondre" avec la requête elle-même, quel que soit leur contenu.
Cela permet d'éviter 99.9 % des failles de type Injection SQL.

Documentation PHP sur PDO : Lien 55

Pour ceux qui veulent savoir ce qu'est le 0.01 % restant (Slide 31) : Lien 56

6.5. L'extension Filter (>= PHP 5.2.0)

Depuis la version 5.2.0, PHP nous fournit l'extension Filter (et activée par défaut !).

La documentation est assez bien faite : Lien 57
Filter permet de transformer une donnée dans le format URL, échappement de quotes (') et de guillemets(").
Cependant, l'utilisateur a la possibilité d'étendre Filter avec ses propres méthodes de formatage.

À noter la possibilité de valider toutes les données d'un formulaire en une seule fois, ce qui est assez propre et lisible (exemple sur la page de la méthode filter_input_array : Lien 58).

Filter permet également de :
- supprimer certains caractères d'une chaîne ;
- vérifier le format d'une chaîne (alphanumérique, mail, IP, etc.).

Il existe aussi une traduction d'un article sur Filter sur developpez.com : Lien 59.

6.6. In.class (PHP 5.3.0)

Les solutions précédentes ne me convenant pas, j'ai développé une classe qui permet de formater facilement les données.

Mon cahier des charges était le suivant :
- facile d'utilisation => l'API est intuitive ;
- concis => API réduite au minimum avec l'appel dynamique sur les objets ;
- gérant les magic_quotes => la première version datait de 2007. C'est inutile maintenant mais je ne l'ai pas supprimée ;
- performant => les données formatées sont mises en cache, donc une même variable ne sera jamais formatée deux fois dans la même page.

Je diffuse maintenant la dernière version du code, après l'avoir réécrit avec les nouveautés de PHP 5.3.0 (Late Static Binding notamment).

Exemples d'utilisation :

```
echo htmlentities($_GET['variable'], ENT_QUOTES);
echo mysql_real_escape_string($maVariableLocale);
```

deviennent

```
$In = In::getInstance(); // en début de page ou de méthode
echo $In->G->HTML->variable;
echo $In->SQL($maVariableLocale);
```

Plus d'exemples et d'explications dans l'en-tête de la classe disponible ici :

Voir/Télécharger In.class.php : Lien 60

6.7. Résumé

Le tableau ci-dessous résume les avantages / inconvénients de chaque solution :

Solution	Interdire	Tout échapper	Filter	In.class
Généralités				
Version de PHP minimum	Toutes	Toutes	5.2.0	5.3.0
Code propre	Non	Non	Oui	Oui
Code concis	Non	Non	Non	Oui
Natif PHP	Oui	Oui	Oui	Non
Extensible	N/A	N/A	Oui	Oui
Gestion des magic_quotes	Non	Non	Non	Oui
Formatage multiple	Non	Non	Oui	Non
Formats supportés nativement				
Raw	Oui	Oui	Oui	Oui
Html	N/A	N/A	Non	Oui
URL	N/A	N/A	Oui	Oui
Base64	N/A	N/A	Non	Oui

Échappement '	N/A	N/A	Oui	Oui
Échappement "	N/A	N/A	Oui	Oui
Échappement SQL	N/A	N/A	Non	Oui
Appel système	N/A	N/A	Non	Non
Expression régulière	N/A	N/A	Non	Non

7. Tests

7.1. Chaîne de test

Lorsque vous voulez tester un formulaire, une URL ou une variable contre tout ce qui a été expliqué au-dessus, vous pouvez utiliser la chaîne de caractères suivante :

```
a'a"a<u>a</u>a/a\a$a;a`a#a
```

Si le comportement est anormal, ou que la chaîne est modifiée à l'affichage, c'est que les données sont mal gérées.

Il est très rare que les sites gèrent l'intégralité de ces caractères, surtout dans les champs "login/password" d'un site.
Le champ password est le plus délicat, car s'il est transformé avec un hash MD5 ou autre, il convient de vérifier qu'il subit les mêmes transformations lors de la saisie et lors de la vérification, avant la transformation MD5.

Retrouvez l'article de Fladnag en ligne : Lien 61

Comment créer des diagrammes avec Flex

Traduction de l'article How to make charts with flex 3 de Zach Dunn paru sur Built Internet : Lien 62.

1. Introduction

Avec Flex, coder en Actionscript est devenu moins pénible et est même devenu un atout pour la conception. Une des meilleures démonstrations de ceci est la simplicité de mise en œuvre des cartes et autres graphiques avec le MXML.

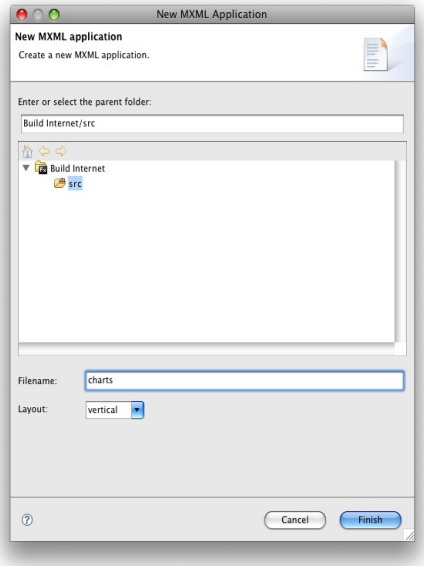
Dans ce tutoriel, nous utiliserons les résultats d'un test issu d'un fichier XML et nous créerons des histogrammes dynamiques en fonction des données en un rien de temps. À la fin de ce cours, vous allez vous impressionner.

2. Étape 1 - Une note, une offre et une installation

Comme vous allez vite vous en rendre compte, je n'utiliserai pas le SDK pour générer les fichiers Flex 3. Bien que ce tutoriel ait été écrit en partant du principe que vous utilisez Flex Builder, le code sous-jacent reste valable. Si vous êtes utilisateurs du SDK, vous devrez modifier voire ajouter quelques étapes pour obtenir le même résultat. Par chance, je n'utiliserai pas le mode conception dans ce tutoriel, profitez-en.

Si vous dépendez du secteur de l'enseignement et si vous n'avez pas encore profité de l'offre Flex 3 gratuit pour l'éducation (Lien 63), n'hésitez pas et faites-le. Cela vous aidera énormément pour le développement d'applications Flex.

Démarrez Flex 3 builder et créez une nouvelle application MXML (version française : Fichier > Nouveau > Application MXML; version anglaise : File > New > MXML Application). La fenêtre ci-dessous apparaîtra. Donnez à l'application le nom que vous voulez (pour ma part, j'utiliserai 'charts') et définissez la mise en page en 'vertical'. Cela évitera que chaque élément soit positionné en haut du canevas précédent comme c'est le cas par défaut.



3. Étape 2 - Utilisation des fichiers XML

CPour ce tutoriel, nous allons charger les données à partir de deux fichiers XML. Chacun contient les notes des tests d'aptitude de trois étudiants différents. Depuis le répertoire 'src' de votre projet, créez un nouveau dossier appelé 'assets'. Téléchargez les fichiers XML (Lien 64) et importez-les dans le répertoire que vous venez de créer. Voici la structure de base des XML :

```
<chart>
    <student>
        <name>John</name>
        <sat>1300</sat>
    </student>
</chart>
```

Comme vous pouvez le voir nous ne récupérons que deux informations par nœud : le nom de l'étudiant et sa note. Si

vous voulez en savoir plus sur le chargement de données XML avec Flex, je vous recommande de prendre une minute pour regarder ce tutoriel : Lien 65. Sinon, commencez par créer directement sous la balise d'ouverture de l'application, le Service HTTP suivant, pour charger le XML.

```
<mx:HTTPService url="assets/test-data.xml"
id="testData" result="xmlHandler(event)"
resultFormat="e4x"/>
```

Nous envoyons la requête HTTP en utilisant l'événement creationComplete dans la balise de l'application.

```
<mx:Application
xmlns:mx="http://www.adobe.com/2006/mxml"
layout="vertical"
creationComplete="testData.send()">
```

Avec un peu d'actionscript nous allons avoir un XML chargé et prêt à être utilisé. Insérez cette balise script juste en dessous de la balise Application et au-dessus du service HTTP.

```
<mx:Script>
<![CDATA[
import mx.rpc.events.ResultEvent;

//XML List for loaded XML file. Must be bindable!
[Bindable]private var testInfo:XMLList;

private function
xmlHandler(evt:ResultEvent):void{
    //Sets testInfo's root as the student.
Everything else referenced in respect to this.
    testInfo = evt.result.student;
}]]
]]>
</mx:Script>
```

4. Étape 3 - Création du diagramme

Maintenant que nous avons toutes les données sous la main, il serait bien d'en faire quelque chose... Comme l'utiliser dans un diagramme par exemple ! Ajoutez le composant suivant en dessous du tag du Service HTTP.

```
<!--Contains page components. Design only-->
<mx:VBox horizontalAlign="center">

  <!--Panel effects design only-->
  <mx:Panel horizontalAlign="center"
title="College Test Score Data">

    <!--The Chart 'testChart'-->
    <mx:ColumnChart dataProvider="{testInfo}"
id="testChart" showDataTips="true">
      <mx:horizontalAxis>
        <mx:CategoryAxis
dataProvider="{testInfo}" categoryField="name"/>
      </mx:horizontalAxis>
      <mx:series>
        <mx:ColumnSeries xField="name"
yField="sat" displayName="Sat Score"/>
      </mx:series>
    </mx:ColumnChart>

  </mx:Panel>

  <!--Legend for Chart Data-->
  <mx:Legend dataProvider="{testChart}"/>

</mx:VBox>
```

Si ça a l'air compliqué, il n'en est rien. Voici une explication détaillée :

1. le VBox est utilisé comme conteneur et permet de centrer les composants qu'il contient. Il n'est utile que pour faciliter la mise en page ;
2. le Panel est utilisé pour contenir le diagramme et pour afficher le titre. Là encore ce composant facilite la mise en page ;
3. le ColumnChart commence la définition d'un composant diagramme. Notez qu'il existe plusieurs types de graphique (Camembert, Histogramme...) et la plupart fonctionnent de la même manière que le diagramme. Adobe a une bonne documentation sur ce sujet : Lien 66. Le champ dataProvider est lié à la liste XML 'testInfo' que nous avons créée précédemment et qui contient les données XML. Pour plus d'informations sur l'attachement de données, voici un tutoriel qui explique l'essentiel : Lien 67. Le Data tips est activé et affichera une info bulle lors du passage de la souris sur une colonne ;
4. l'horizontalAxis permet de définir les informations affichées sur l'axe des abscisses. Le CategoryAxis permet de spécifier la source des données et le champ à utiliser ;
5. la balise serie contient les informations qui seront utilisées pour créer le diagramme ;
6. le ColumnSeries permet de spécifier le champ du XML qui sera affiché sur chaque axe. Le displayName définit le libellé des informations de la colonne. Bien qu'il n'y ait qu'une seule colonne dans cet exemple, en ajouter de nouvelles revient à ajouter de nouvelles columnSeries dans la Serie ;
7. la legend dépend du composant ColumnChart (appelé testChart) et affichera la valeur du displayName de chaque ColumnSeries ainsi que la couleur.

5. Étape 4 - Intervertir les données

Comme vous le savez, nous avons deux fichiers XML contenant différentes données utilisables pour ce diagramme. Actuellement, nous avons uniquement chargé le premier (test-data.xml). Cet exemple va vous montrer la facilité avec laquelle un graphique Flex peut s'adapter dynamiquement aux contenus. Pour ce faire, nous devons effectuer quelques modifications sur le code que nous venons d'écrire. On va commencer par ajouter deux boutons directement sous le ColumnChart, qui nous serviront à changer le fichier XML utilisé sur le graphique. Je les ai créés dans un Hbox pour faciliter la mise en page.

```
<mx:HBox height="30" horizontalAlign="center">
      <mx:Button  label="Data Set 1"
click="changeData('set1')"/>
      <mx:Button label="Data Set 2"
click="changeData('set2')"/>
</mx:HBox>
```

Ensuite, nous devons créer la fonction pour changer la

source des données. Nous venons de préparer le terrain avec les boutons que nous venons de déclarer. En effet, sur l'événement de clic, on passe le jeu de données XML choisi à la fonction appelée ChangeData. Nous allons créer celle-ci en dessous de la fonction xmlHandler dans la balise script.

```
private function changeData(dataSet:String):void{
    //Determine which set should be loaded
    switch (dataSet){
        case ('set1'):
            //Set URL target to Test Data 1
            testData.url ="assets/test-data.xml";
            break;
        case ('set2'):
            //Set URL target to Test Data 2
            testData.url ="assets/test-
data2.xml";
            break;
        default:
            //If somehow it's neither, just leave
it be
            break;
    }
    //Send out new URL Request to refresh chart
    testData.send();
}
```

Le code ci-dessus est un simple commutateur en actionscript qui évalue la variable passée en paramètre et affecte l'emplacement du fichier XML correspondant, à l'attribut URL du HTTPService. Une fois affectée, nous devons envoyer la requête via la méthode send pour rafraichir les données.

La partie sympa sur l'attachement de données est que nous n'avons pas besoin de modifier la structure du graphique. Le diagramme est défini pour afficher automatiquement les valeurs de testInfo. Avec cette fonction, nous chargeons simplement différentes données dans testInfo, et Flex les récupère de là. Foncez et essayez-la maintenant. Si tout se passe comme prévu, vous devriez voir le graphique changer lorsque vous appuyez sur le bouton. Ça semble plutôt pas mal, mais nous allons faire une dernière modification pour accentuer les changements sur le diagramme.

6. Étape 5 - Animer la transition

Les diagrammes Flex sont plus classe lorsque nous ajoutons un peu d'animation. Nous allons en utiliser une simple pour animer la transition de données sur le graphique. Au-dessus de la balise HTTPService, ajoutez la balise ci-dessous :

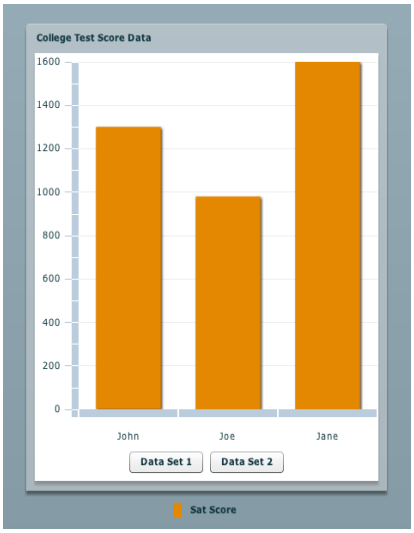
```
<mx:SeriesInterpolate id="changeEffect"
duration="2000"/>
```

Cela crée une belle transition entre les données. Nous lui donnons un id pour y faire référence à d'autres endroits, et lui affectons une durée de deux secondes. Pour utiliser cette animation, ajoutez l'attribut showDataEffect à la balise ColumnSeries et liez-la à la balise SeriesInterpolate. Cela aura pour effet d'appeler l'effet de transition chaque fois que de nouvelles données seront affichées sur le graphique.

```
<mx:ColumnSeries showDataEffect="{changeEffect}"
xField="name" yField="sat" displayName="Sat
Score"/>
```

7. Étape 6 - Le graphique terminé

Allez, sauvegardez ce que vous avez fait et regardez le résultat. Si vous avez bien tout suivi, vous devriez voir quelque chose comme l'exemple ci-dessous avec la transition lorsque vous appuyez sur les boutons. Pas mal, hein ?



S'il y a eu un problème en chemin, j'ai inclus les fichiers sources pour le corriger. N'oubliez pas d'essayer les autres types de graphique. Maintenant vous avez ajouté une nouvelle connaissance sur Flex. Félicitations !

Téléchargez le fichier source : Lien 68.
Regardez la démo en ligne : Lien 69.

Retrouvez l'article de Zach Dunn traduit par KalyParker en ligne : Lien 70

Web sémantique et (X)HTML5 : les microdonnées et les éléments sémantiques

HTML5 est le standard du Web à venir. Il s'oriente donc vers ce qui semble actuellement être le futur des pages Web : il n'y aura pas que du multimédia, il y aura aussi du sémantique. Deux voies sont explorées dans ce standard : l'utilisation de balises sémantiques pour structurer la page et les microdonnées.

1. Introduction

Une des caractéristiques principales de HTML5 est qu'il ne cassera rien au support existant pour les versions précédentes. Tout le code HTML4 continuera de fonctionner. Si on veut faire mieux, il suffira d'introduire les doses de HTML5 là où c'est nécessaire. Rien ne doit être modifié pour utiliser ces nouvelles fonctionnalités, HTML5 se base sur sa version précédente. Par exemple, pour ajouter un peu de Web sémantique à des pages HTML4 déjà existantes, il suffit de changer le doctype du document et de se laisser aller à ajouter les quelques balises et attributs qu'il faut.

Commençons tout d'abord par modifier le doctype. En HTML5, il est très simple, il est seul et unique, il ne fait que quinze octets dans les encodages sur huit bits :

```
<!DOCTYPE html>
```

2. Les éléments sémantiques

HTML5 apporte le support de nouveaux éléments pour aider le navigateur à déterminer la sémantique d'une page. Ces éléments sont faciles à comprendre et à assimiler pour tout qui connaît un peu d'anglais. Passons-les en revue, car les microdonnées peuvent s'ajouter à ces balises.

2.1. Balises

Balise	Signification
<section>	Une section représente un document générique ou une section d'une application. Par exemple, il pourrait s'agir d'un titre et du texte subordonné, comme toute section du présent article.
<nav>	Il s'agit d'une section particulière d'une page, celle qui contient les liens dits de navigation. Il n'est pas nécessaire de mettre tous les liens d'une page dans une telle balise, seuls les plus gros blocs de liens devraient y trouver place.
<article>	Une partie qui se suffit à elle-même d'une page, qui pourrait être redistribuée telle quelle, sans ce qui l'entoure, comme un post sur un blog ou un forum.
<aside>	Des éléments qui sont reliés au contenu mais sans en faire partie trouvent place dans ces balises. Dans un livre, cela correspond au texte mis dans les colonnes sur les côtés, par exemple.
<hgroup>	L'en-tête d'une section, utilisé pour grouper des éléments h1 à h6 quand la page possède plusieurs niveaux de titres.
<header>	Des aides à la navigation, tout l'en-tête du site. On attend généralement que cet élément contienne le titre principal h1 de la page, mais ce n'est pas requis. On peut aussi y placer un espace de recherche ou la table des matières de la page.
<footer>	Le bas de la page. Par exemple, on pourrait placer dans une telle balise la fin de cette page, avec un lien vers le forum, une page de contact, le copyright, etc.
<time>	Une heure sur une horloge à 24 heures, une date précise dans le calendrier grégorien.
<mark>	Une portion de texte marquée pour référence ultérieure.

2.2. Exemples

Cette suite de descriptions peut paraître assez difficile à intégrer, à comprendre, d'où ces quelques exemples.

2.2.1. L'en-tête

On va se baser sur l'en-tête simplifié d'un blog pour cette section.

Avant

```
<div id="header">
  <h1>Blog de Tartempion</h1>
  <p class="pun">Voici le titre de mon blog. </p>
  <ul>
    <li><a href="/">Accueil</a></li>
    <li><a href="/author/">L'auteur</a></li>
  </ul>
</div>
```

Ce code est tout à fait valide en HTML5 ! Cependant, le standard définit des balises beaucoup plus précises pour exprimer la signification du contenu.

2.2.1.1. <header>

Pour indiquer ce qu'est l'en-tête d'un blog, avec un titre, un sous-titre et un menu, l'opération est très simple.

```
<header>
  <h1>Blog de Tartempion</h1>
  <p class="pun">Voici le titre de mon blog. </p>
  <ul>
```

```
    <li><a href="/">Accueil</a></li>
    <li><a href="/author/">L'auteur</a></li>
  </ul>
</header>
```

2.2.1.2. <hgroup>

Reste cependant le sous-titre... On ne peut pas le mettre tel quel en titre de niveau deux, ni mettre les articles du blog en tant que titres de niveau trois. La solution ? <hgroup>.

```
<header>
  <hgroup>
    <h1>Blog de Tartempion</h1>
    <h2>Voici le titre de mon blog. </h2>
  <hgroup>
  <ul>
    <li><a href="/">Accueil</a></li>
    <li><a href="/author/">L'auteur</a></li>
  </ul>
</header>
```

Cette balise permet d'encadrer plusieurs titres liés ; cela signifie que, si l'on devait réaliser une structure arborescente de la page, ces deux titres seraient dans la même case et subordonneraient tous les titres de la page, qu'ils soient de niveau deux ou inférieur.

2.2.2. Le contenu

Cette fois, prenons pour exemple un article simplifié du blog.

Avant
```
<div class="post">
  <p class="date">Posté le premier avril 2011</p>
  <h2>Mon article</h2>
  <p>
    Lorem ipsum dolor.
  </p>
</div>
```

2.2.2.1. <article>

Tout d'abord, on peut exprimer que le div contient ici un article.

```
<article>
  <header>
    <p class="date">Posté le premier avril
2011</p>
    <h1>Mon article</h1>
  </header>
  <p>
    Lorem ipsum dolor.
  </p>
</article>
```

Il faut cependant ici remarquer que la balise <h2> s'est muée en <h1> et s'est retrouvée dans un élément <header>. En effet, cette dernière balise veut rassembler tous les éléments de l'en-tête d'un article.

Ensuite, on remarque aisément ce qui choquera bon nombre de personnes habituées à HTML4 : la présence de plusieurs <h1> par page !

En effet, chaque article est une nouvelle section dans la page ; en tant que tel, il possède un titre de niveau un. Dans la représentation de la page, on pourrait voir chaque article comme une petite page, possédant elle aussi son en-tête, son pied de page... et son titre de niveau un.

Pourquoi diable un tel changement ? Beaucoup de pages sont générées à l'aide de systèmes de templates. Cette méthode permet de simplifier fortement les algorithmes sous-jacents des frameworks Web (plus besoin de se rappeler à quel niveau s'arrête le template parent, il suffit que chaque article existe avec ses titres qui sont cohérents s'il devait être affiché seul).

2.2.2.2. <time>

Dans cet en-tête, on a défini une date, que nous interprétons comme la date de publication. Il faudrait l'indiquer clairement au navigateur. Ensuite, on peut l'écrire dans n'importe quelle langue, le navigateur doit pouvoir l'interpréter correctement. D'où le changement suivant :

```
<article>
  <header>
    <time datetime="2011-04-01" pubdate>Posté le
premier avril 2011</time>
    <h1>Mon article</h1>
  </header>
  <p>
    Lorem ipsum dolor.
  </p>
</article>
```

On aurait aussi pu définir une heure de publication au format 24 heures avec un décalage horaire :

```
<article>
  <header>
    <time datetime="2011-04-01T12:42:42+02:00"
pubdate>Posté le premier avril 2011</time>
    <h1>Mon article</h1>
  </header>
  <p>
    Lorem ipsum dolor.
  </p>
</article>
```

Il faut faire attention à l'emplacement de cette balise de temps : dans un élément article, elle définit la date de publication de l'article ; en dehors, elle définit la date de publication du document dans son entièreté.

2.2.3. La navigation

Ici, on se basera sur une simple liste d'éléments définissant le menu d'un site.

Avant
```
<div id="tabs">
  <ul>
    <li><a href="/">Accueil</a></li>
    <li><a href="/author/">L'auteur</a></li>
  </ul>
</div>
```

2.2.3.1. <nav>

On peut aider notamment les personnes avec des déficiences fonctionnelles à utiliser le site en signalant aux logiciels spécifiques où se situe le menu :

```
<nav>
  <ul>
    <li><a href="/">Accueil</a></li>
    <li><a href="/author/">L'auteur</a></li>
  </ul>
</nav>
```

2.2.4. Le pied de page

Aujourd'hui, on distingue deux tendances dans les pieds de page : soit extrêmement légers (une notice de copyright et assez peu d'informations en tout), soit extrêmement lourds (avec plusieurs colonnes de liens).

Le principe est le même dans les deux cas : il faut encapsuler le pied de page dans la balise associée, <footer>. Ensuite, pour des pieds de page plus lourds, vient l'étape d'une subdivision, pour indiquer ce que signifie chaque colonne. Pour ce faire, on se basera sur le pied de page du W3C :

Avant

```
<div id="w3c_footer">
  <div class="w3c_footer-nav">
    <h3>Navigation</h3>
    <ul>
      <li><a href="/">Home</a></li>
      <li><a
href="/standards/">Standards</a></li>
      <li><a
href="/participate/">Participate</a></li>
      <li><a
href="/Consortium/membership">Membership</a></li>
      <li><a href="/Consortium/">About
W3C</a></li>
    </ul>
  </div>
  <div class="w3c_footer-nav">
    <h3>Contact W3C</h3>
    <ul>
      <li><a
href="/Consortium/contact">Contact</a></li>
      <li><a href="/Help/">Help and FAQ</a></li>
      <li><a
href="/Consortium/sup">Donate</a></li>
      <li><a href="/Consortium/siteindex">Site
Map</a></li>
    </ul>
  </div>
  <div class="w3c_footer-nav">
    <h3>W3C Updates</h3>
    <ul>
      <li><a
href="http://twitter.com/W3C">Twitter</a></li>
      <li><a
href="http://identi.ca/w3c">Identi.ca</a></li>
    </ul>
  </div>
  <p class="copyright">Copyright © 2009 W3C</p>
</div>
```

2.2.4.1. <footer>

Tout d'abord, il faut encadrer tout ça dans un pied de page bien défini :

```
<footer>
  <div class="w3c_footer-nav">
    <h3>Navigation</h3>
    <ul>
      <li><a href="/">Home</a></li>
      <li><a
href="/standards/">Standards</a></li>
      <li><a
href="/participate/">Participate</a></li>
      <li><a
href="/Consortium/membership">Membership</a></li>
      <li><a href="/Consortium/">About
W3C</a></li>
    </ul>
  </div>
  <div class="w3c_footer-nav">
    <h3>Contact W3C</h3>
    <ul>
      <li><a
href="/Consortium/contact">Contact</a></li>
      <li><a href="/Help/">Help and FAQ</a></li>
      <li><a
href="/Consortium/sup">Donate</a></li>
      <li><a href="/Consortium/siteindex">Site
Map</a></li>
    </ul>
  </div>
  <div class="w3c_footer-nav">
    <h3>W3C Updates</h3>
    <ul>
      <li><a
href="http://twitter.com/W3C">Twitter</a></li>
      <li><a
href="http://identi.ca/w3c">Identi.ca</a></li>
    </ul>
  </div>
  <p class="copyright">Copyright © 2009 W3C</p>
</footer>
```

2.2.4.2. <nav> et <section>

Ensuite, on remarque qu'il y a deux types de colonnes : les deux premières donnent des liens de navigation et entrent donc dans des <nav> ; la dernière définit quelques liens externes liés au site et entre donc dans un <section>.

On modifie également les niveaux des titres, comme pour les articles.

```
<footer>
  <nav>
    <h1>Navigation</h1>
    <ul>
      <li><a href="/">Home</a></li>
      <li><a
href="/standards/">Standards</a></li>
      <li><a
href="/participate/">Participate</a></li>
      <li><a
href="/Consortium/membership">Membership</a></li>
      <li><a href="/Consortium/">About
W3C</a></li>
    </ul>
  </nav>
  <nav>
    <h1>Contact W3C</h1>
```

```
    <ul>
      <li><a
href="/Consortium/contact">Contact</a></li>
      <li><a href="/Help/">Help and FAQ</a></li>
      <li><a
href="/Consortium/sup">Donate</a></li>
      <li><a href="/Consortium/siteindex">Site
Map</a></li>
    </ul>
  </nav>
  <section>
    <h1>W3C Updates</h1>
    <ul>
      <li><a
href="http://twitter.com/W3C">Twitter</a></li>
      <li><a
href="http://identi.ca/w3c">Identi.ca</a></li>
    </ul>
  </section>
  <p class="copyright">Copyright © 2009 W3C</p>
</footer>
```

2.3. En résumé

Voici donc ce à quoi pourrait ressembler votre page de blog :

```
<!doctype html>
<html>
  <head>
    <title>Mon blog</title>
  </head>
  <body>
    <nav>
      <ul>
        <li><a href="/">Accueil</a></li>
        <li><a href="/author/">L'auteur</a></li>
      </ul>
    </nav>
    <header>
      <hgroup>
        <h1>Blog de Tartempion</h1>
        <h2>Voici le titre de mon blog. </h2>
      <hgroup>
      <ul>
        <li><a href="/">Accueil</a></li>
        <li><a href="/author/">L'auteur</a></li>
      </ul>
    </header>
    <article>
      <header>
        <time datetime="2011-04-
01T12:42:42+02:00" pubdate>Posté le premier avril
2011</time>
        <h1>Mon article</h1>
      </header>
      <p>
        Lorem ipsum dolor.
      </p>
    </article>
    <footer>
      <ul>
        <li><a href="/">Accueil</a></li>
        <li><a href="/author/">L'auteur</a></li>
      </ul>
    </footer>
  </body>
</html>
```

2.4. Le cas XHTML5

Il est aussi possible d'utiliser un formalisme XML pour le nouveau standard HTML5. La spécification XHTML5 n'apporte rien par rapport aux spécifications de XML, il suffit donc d'écrire ses documents XHTML5 comme des documents XHTML1.

La principale différence, cependant, est qu'il n'est pas nécessaire de mettre un DOCTYPE (Lien 71) :

```
XML documents may contain a DOCTYPE if desired,
but this is not required to conform to this
specification.
```

Le reste de la spécification n'est pas intéressant dans notre cas.

On remarque aussi que de nouvelles choses sont possibles en HTML5 et totalement réprimandées en XML, notamment :

```
<time datetime="2011-04-01T12:42:42+02:00"
pubdate>...</time>
```

On procédera donc avec le même artifice qu'en XHTML1 :

```
<time datetime="2011-04-01T12:42:42+02:00"
pubdate="pubdate">...</time>
```

On aura donc un code comme celui-ci :

```
<!doctype html>
<html>
  <head>
    <title>Mon blog</title>
  </head>
  <body>
    <nav>
      <ul>
        <li><a href="/">Accueil</a></li>
        <li><a href="/author/">L'auteur</a></li>
      </ul>
    </nav>
    <header>
      <hgroup>
        <h1>Blog de Tartempion</h1>
        <h2>Voici le titre de mon blog. </h2>
      <hgroup>
      <ul>
        <li><a href="/">Accueil</a></li>
        <li><a href="/author/">L'auteur</a></li>
      </ul>
    </header>
    <article>
      <header>
        <time datetime="2011-04-
01T12:42:42+02:00" pubdate="pubdate">Posté le
premier avril 2011</time>
        <h1>Mon article</h1>
      </header>
      <p>
        Lorem ipsum dolor.
      </p>
    </article>
    <footer>
      <ul>
        <li><a href="/">Accueil</a></li>
```

```
        <li><a href="/author/">L'auteur</a></li>
      </ul>
    </footer>
  </body>
</html>
```

Vous remarquerez la présence d'un DOCTYPE. Ceci permet de valider le document par les outils mis à disposition par le W3C, le fait qu'il s'agisse d'un document XHTML5 ne dérange nullement le valideur.

3. Les microdonnées

On vient de voir toute une série d'éléments nouveaux dans HTML5. Cependant, ces nouveaux éléments, aussi sémantiques soient-ils, ne permettent pas de tout définir. Comment décrire une personne, comme on peut le faire en RDFa (Lien 72) ? Peut-on créer une balise <person> ? Ce n'est pas envisageable. Il faut donc regarder à côté pour trouver une autre manière de faire.

Les microdonnées annotent l'arbre DOM avec des paires nom-valeur de vocabulaires personnalisés avec une certaine portée.

Ici, remarquons l'utilisation du terme *vocabulaire* : pour définir des microdonnées, il faut utiliser certains vocabulaires qui donneront tout leur sens aux annotations.

Aussi, ces microdonnées sont définies avec une certaine portée : tous les enfants d'une certaine balise utiliseront le même vocabulaire, pas les frères ou sœurs de cette balise, qui pourront utiliser d'autres vocabulaires, à leur guise.

Ces microdonnées sont une manière d'ajouter des informations à la base HTML existante, elles fournissent un vocabulaire plus riche et plus précis que ce que peut fournir le HTML.

La définition des vocabulaires a bien évidemment été normalisée : Lien 73. Google met à disposition une liste des vocabulaires déjà définis : Lien 74.

3.1. Valeurs des propriétés

Ajouter des microdonnées à un texte revient à ajouter des attributs aux éléments HTML déjà existants. En général, la valeur de ces propriétés correspond au contenu texte (pas les autres balises, donc). Voici un tableau récapitulatif à ce sujet, avec les exceptions pour lesquelles la valeur n'est pas le contenu textuel.

Élément	Attribut pris pour valeur
<meta>	content
<audio>, <embed>, <iframe>, <img>, <source>, <video>	src
<a>, <area>, <link>	href
<object>	data
<time>	datetime

Tous les autres éléments voient leur contenu textuel pris pour valeur. Par exemple,

```
<h1>Mon titre</h1>
```

aura pour valeur Mon titre.

3.2. Description d'une personne

Pour décrire une personne, on utilisera le vocabulaire http://data-vocabulary.org/Person.

```
<section>
  <h1>Tartempion</h1>
  <p><img src="http://tarte.net/photo.jpg"></p>
  <p><a href="http://tarte.net/">Mon site</a></p>
</section>
```

On définira le champ d'action de ce vocabulaire comme étant l'élément courant et ses enfants. Seul le contenu de la balise <section> doit être marqué, c'est donc lui qui hérite de ces attributs.

```
<section itemscope itemtype="http://data-
vocabulary.org/Person">
  <h1>Tartempion</h1>
  <p><img src="http://tarte.net/photo.jpg"></p>
  <p><a href="http://tarte.net/">Mon site</a></p>
</section>
```

Ensuite, il faut marquer chaque élément pour indiquer sa sémantique propre, à l'aide d'attributs itemprop :

```
<section itemscope itemtype="http://data-
vocabulary.org/Person">
  <h1 itemprop="name">Tartempion</h1>
  <p><img itemprop="photo"
src="http://tarte.net/photo.jpg"></p>
  <p><a itemprop="url"
href="http://tarte.net/">Mon site</a></p>
</section>
```

Que contient d'autre ce vocabulaire ?

Propriété	Signification
name	Nom
nickname	Surnom
photo	Photo
title	Titre
role	Rôle
url	Page Web
association	Association à une organisation (employeur, par exemple)
friend, contact, acquaintance	Relation sociale avec une autre personne
address	Localisation (peut posséder les sous-propriétés street-address (adresse), locality (localité), region (région), postal-code (code postal), country-name (pays))

L'adresse est un autre élément de microdonnées ! Il faut donc utiliser un vocabulaire spécifique (http://data-vocabulary.org/Address).

```
<dl>
        <dt>Adresse</dt>
        <dd itemprop="address"> itemscope
itemtype="http://data-vocabulary.org/Address">
                <span itemprop="street-
address'>Rue de Mouscron, 42</span><br>
                <span itemprop="postal-
code">1042</span>
                <span
itemprop="locality">Bruxelles</span><br/>
                <span itemprop="country-
name">Belgium</span>
        </dd>
</dl>
```

Remarquez que l'on n'est pas obligé de spécifier tous les champs d'une adresse, seulement ceux qui sont utiles dans ce cas. De même, tous les pays ne suivent pas forcément ce type de format d'adresses ; cependant, il devrait être utilisable dans la très grande majorité des cas.

3.3. Description d'une société

Là aussi, un vocabulaire est prévu. Il s'agit de http://data-vocabulary.org/Organization.

Propriété	Signification
name	Nom
url	Page Web
address	Adresse de la société, comme pour une personne
tel	Numéro de téléphone
geo	Coordonnées géographiques, avec latitude et longitude

Prenons pour exemple Google. En HTML5, avant les microdonnées, la page *à propos* pourrait contenir ceci :

```
<article>
  <h1>Google, Inc.</h1>
  <p>
    1600 Amphitheatre Parkway<br>
    Mountain View, CA 94043<br>
    USA
  </p>
  <p>650-253-0000</p>
  <p><a
href="http://www.google.com/">Google.com</a></p>
</article>
```

Commençons par définir le vocabulaire utilisé.

```
<article itemscope itemtype="http://data-
vocabulary.org/Organization">
  <h1>Google, Inc.</h1>
  <p>
    1600 Amphitheatre Parkway<br>
    Mountain View, CA 94043<br>
    USA
  </p>
  <p>650-253-0000</p>
  <p><a
href="http://www.google.com/">Google.com</a></p>
</article>
```

Cette société a un nom.

```
<article itemscope itemtype="http://data-
vocabulary.org/Organization">
  <h1 itemprop="name">Google, Inc.</h1>
  <p>
    1600 Amphitheatre Parkway<br>
    Mountain View, CA 94043<br>
    USA
  </p>
  <p>650-253-0000</p>
  <p><a
href="http://www.google.com/">Google.com</a></p>
</article>
```

Le premier paragraphe en définit l'adresse.

```
<article itemscope itemtype="http://data-
vocabulary.org/Organization">
  <h1 itemprop="name">Google, Inc.</h1>
  <p itemprop="address" itemscope
itemtype="http://data-vocabulary.org/Address">
    1600 Amphitheatre Parkway<br>
    Mountain View, CA 94043<br>
    USA
  </p>
  <p>650-253-0000</p>
  <p><a
href="http://www.google.com/">Google.com</a></p>
</article>
```

Maintenant, il faut définir chaque champ de l'adresse.

```
<article itemscope itemtype="http://data-
vocabulary.org/Organization">
  <h1 itemprop="name">Google, Inc.</h1>
  <p itemprop="address" itemscope
itemtype="http://data-vocabulary.org/Address">
    <span itemprop="street-address">1600
Amphitheatre Parkway</span><br>
    <span itemprop="locality">Mountain
View</span>, <span itemprop="region">CA</span>
<span itemprop="postal-code">94043</span><br>
    <span itemprop="country-name">USA</span>
  </p>
  <p>650-253-0000</p>
  <p><a
href="http://www.google.com/">Google.com</a></p>
</article>
```

Pour les informations téléphoniques, la procédure n'est pas très différente. Remarquons que l'on sort de la balise de paragraphe pour laquelle on a utilisé le vocabulaire Address, on retourne donc au vocabulaire Organization. De même pour le site Web.

```
<article itemscope itemtype="http://data-
vocabulary.org/Organization">
  <h1 itemprop="name">Google, Inc.</h1>
  <p itemprop="address" itemscope
itemtype="http://data-vocabulary.org/Address">
    <span itemprop="street-address">1600
Amphitheatre Parkway</span><br>
    <span itemprop="locality">Mountain
View</span>, <span itemprop="region">CA</span>
<span itemprop="postal-code">94043</span><br>
    <span itemprop="country-name">USA</span>
  </p>
  <p itemprop="tel">650-253-0000</p>
  <p itemprop="url"><a
href="http://www.google.com/">Google.com</a></p>
</article>
```

On peut évidemment définir plusieurs numéros de téléphone, plusieurs sites Web. Il suffit de définir les éléments sémantiques à l'aide de span et non dans un paragraphe :

```
<article itemscope itemtype="http://data-vocabulary.org/Organization">
  <h1 itemprop="name">Google, Inc.</h1>
  <p itemprop="address" itemscope itemtype="http://data-vocabulary.org/Address">
    <span itemprop="street-address">1600 Amphitheatre Parkway</span><br>
    <span itemprop="locality">Mountain View</span>, <span itemprop="region">CA</span> <span itemprop="postal-code">94043</span><br>
    <span itemprop="country-name">USA</span>
  </p>
  <p>
    US: <span itemprop="tel">650-253-0000</span><br>
    UK: <span itemprop="tel">00-1-650-253-0000</span>
  </p>
  <p itemprop="url"><a href="http://www.google.com/">Google.com</a></p>
</article>
```

Dernier élément : la géolocalisation, définie elle aussi dans son propre vocabulaire, http://data-vocabulary.org/Geo. On peut, par exemple, l'utiliser comme ceci :

```
<article itemscope itemtype="http://data-vocabulary.org/Organization">
  <h1 itemprop="name">Google, Inc.</h1>
  <p itemprop="address" itemscope itemtype="http://data-vocabulary.org/Address">
    <span itemprop="street-address">1600 Amphitheatre Parkway</span><br>
    <span itemprop="locality">Mountain View</span>, <span itemprop="region">CA</span> <span itemprop="postal-code">94043</span><br>
    <span itemprop="country-name">USA</span>
  </p>
  <p>
    US: <span itemprop="tel">650-253-0000</span><br>
    UK: <span itemprop="tel">00-1-650-253-0000</span>
  </p>
  <p itemprop="url"><a href="http://www.google.com/">Google.com</a></p>
  <span itemprop="geo" itemscope itemtype="http://data-vocabulary.org/Geo">
    <meta itemprop="latitude" content="37.4149" />
    <meta itemprop="longitude" content="-122.078" />
  </span>
</article>
```

3.4. En XHTML5

Comme précédemment, peu de choses sont à changer. En guise d'exemple, voici le dernier snippet adapté pour XHTML5 :

```
<article itemscope itemtype="http://data-vocabulary.org/Organization">
  <h1 itemprop="name">Google, Inc.</h1>
  <p itemprop="address" itemscope="itemscope" itemtype="http://data-vocabulary.org/Address">
    <span itemprop="street-address">1600 Amphitheatre Parkway</span><br>
    <span itemprop="locality">Mountain View</span>, <span itemprop="region">CA</span> <span itemprop="postal-code">94043</span><br>
    <span itemprop="country-name">USA</span>
  </p>
  <p>
    US: <span itemprop="tel">650-253-0000</span><br>
    UK: <span itemprop="tel">00-1-650-253-0000</span>
  </p>
  <p itemprop="url"><a href="http://www.google.com/">Google.com</a></p>
  <span itemprop="geo" itemscope="itemscope" itemtype="http://data-vocabulary.org/Geo">
    <meta itemprop="latitude" content="37.4149" />
    <meta itemprop="longitude" content="-122.078" />
  </span>
</article>
```

4. Conclusion

Il ne s'agit ici que d'une introduction, donnant les principes nécessaires à l'utilisation de tous les vocabulaires disponibles.

Retrouvez l'article de Thibaut Cuvelier en ligne : Lien 75

Schema.org : une initiative pour que les moteurs de recherche comprennent les sites Web

Dans le Web tel qu'on le connaît actuellement, les moteurs de recherche sont à peu près incapables de comprendre le contenu des pages Web qu'ils indexent. Pour répondre à une requête, ils font une recherche sur base des mots-clés fournis par l'utilisateur sous certaines contraintes. Si on leur pose une question, ils vont chercher les mots-clés de la question, ils ne vont pas répondre directement à la question.
Cependant, cela évolue : depuis peu, Google est capable de répondre à certaines questions formulées en anglais, en prenant toutefois une précaution oratoire (« la réponse la plus probable est... »). Pour cela, il faut qu'il puisse comprendre le contenu des pages indexées. C'est une large partie du Web sémantique. Pour aider cela, Schema.org a été lancé par trois des plus grands moteurs de recherche actuels.

1. L'initiative Schema.org

Conjointement, Google (Lien 76), Yahoo (Lien 77) et Bing (Lien 78) annoncent le lancement du site Schema.org (Lien 79), le 2 juin 2011 (Lien 80). L'objectif ? Fournir une série de vocabulaires communs pour structurer les données des pages Web. On doit les rapprocher au concept d'ontologie : ils permettent de représenter les connaissances sur un sujet (on parle d'ontologie dans un contexte purement Web sémantique, tandis qu'on parlera plus de vocabulaire quand il s'agit de structurer les données d'une page Web).

Pour une introduction plus approfondie au Web sémantique et à la notion d'ontologie, Introduction au Web sémantique (Lien 81).

Actuellement, un grand nombre de vocabulaires existent déjà : leur seul problème est de ne jamais être rassemblés. On peut citer Dublin Core Metadata Initiative (Lien 82), The FOAF Project (Lien 83), The DBpedia Ontology (Lien 84) et bien d'autres encore, comme data-vocabulary.org (Lien 85), déjà une initiative Google, ou bien le protocole Open Graph de Facebook (Lien 86). Comment s'y retrouver ? La solution retenue a été de fournir un ensemble de vocabulaires génériques, à même de couvrir l'ensemble des besoins quotidiens (avec les vocabulaires Schema.org, on peut décrire aussi bien une attraction touristique qu'un film ou un concert). Pour ceux qui n'auraient pas assez de vocabulaire à leur disposition, il est toujours possible d'écrire des extensions. Le problème de ces extensions est qu'il n'est pas certain que toutes les applications puissent comprendre ces données supplémentaires.

On peut se demander quel est le lien entre toutes les technologies du Web sémantique (en ce compris SPARQL (Lien 87) ou le RDF). C'est simple : des données structurées sur une page peuvent être transformées en triplets RDF (Lien 88), qui peuvent être exploités aisément.

Ce tutoriel se basera exclusivement sur les techniques de structuration des données disponibles dans le standard HTML5, soit les microdonnées (voir Web sémantique et HTML5 : les microdonnées et les éléments sémantiques pour une introduction : Lien 89). La raison en est simple : tous les exemples du site Schema.org ont été rédigés en ce sens. Il n'est pas cependant interdit d'utiliser RDFa pour exploiter ces vocabulaires (une introduction à RDFa est également disponible : Lien 90).

2. L'équivalence données structurées - triplets RDF

Ce point est spécifiquement discuté par le standard des microdonnées, on va étudier cette partie un peu plus en détail à l'aide d'un exemple simple. Le standard est disponible en tant que *draft* sur le site du consortium W3C : Lien 91. Une section est particulièrement dédiée à la conversion des microdonnées en RDF (Lien 92), en décrivant l'algorithme à suivre pour obtenir tous les triplets de la page. On ne décrira pas ici le contenu de cette section dans tous les détails, certains raccourcis seront pris.

2.1. <title>

La balise <title>, si elle est remplie, donne le triplet suivant :

- sujet : l'adresse du document ;
- prédicat : Lien 93 ;
- objet : le contenu de l'élément <title>.

Si ces triplets RDF ne vous semblent pas clairs, approfondissez Apprendre RDF par l'exemple avec FOAF : Lien 94.

2.2. <a>, <area> et <link>

Ensuite, pour toutes les balises <a>, <area> et <link> du document, on garde celles qui ont un attribut rel défini et href qui pointe vers une URL qui existe. Cela permet d'obtenir une liste de jetons pour chaque élément : on sépare le contenu de l'attribut rel d'un élément en une liste, prenant l'espace comme séparateur, cette liste est la liste de jetons de l'élément.

On distingue ensuite deux types de jetons : ceux qui contiennent un deux-points et ceux qui n'en ont pas. Pour ces seconds, on peut construire le triplet suivant :

- sujet : l'adresse du document ;
- prédicat : Lien 95 suivi du jeton ;
- objet : l'attribut href.

Pour les premiers, par contre :

- sujet : l'adresse du document ;
- prédicat : le jeton ;
- objet : l'attribut href.

On remarque que seule l'interprétation du jeton change, le prédicat aussi par voie de conséquence.

On génère donc un triplet par élément de l'attribut rel, une seule balise peut donc générer un nombre très grand de triplets.

2.3. <meta>

Pour autant que les attributs name et content soient définis, on reprend les mêmes triplets générés que précédemment, avec l'équivalence name-jeton et content-href, avec les différences suivantes :

- une balise ne peut générer qu'un triplet ;
- le contenu ne doit pas forcément être une URL.

2.4. <blockquote> et <q>

Il faut un attribut cite pour qu'on puisse générer un triplet.

- sujet : l'adresse du document ;
- prédicat : Lien 96 ;
- objet : l'URL absolue qui résulte de la résolution de la valeur de l'attribut cite relativement au document.

2.5. Éléments avec microdonnées

On génère les triplets pour chacun des éléments avec microdonnées. Cette partie étant suffisamment complexe, on se limitera ici à un exemple ; les triplets RDF générés seront présentés sous le format Turtle (cela permet une correspondance simple entre le code HTML et les triplets, on pourrait le faire à l'aide d'expressions régulières si on désire rester simpliste).

Un bout de page HTML5 contenant des microdonnées

```
<section itemscope itemtype="http://data-vocabulary.org/Person">
  <h1 itemprop="name">Tartempion</h1>
  <p><img itemprop="photo" src="http://tarte.net/photo.jpg"></p>
  <p><a itemprop="url" href="http://tarte.net/">Mon site</a></p>
</section>
```

Sa retranscription en triplets (Turtle)

```
@prefix dv: <http://data-vocabulary.org/>

<http://tarte.net/moi/> a dv:Person ;
        dv:name "Tartempion" ;
        dv:photo "http://tarte.net/photo.jpg" ;
        dv:url "http://tarte.net/" .
```

3. En guise d'exemple

On va reprendre ici l'exemple officiel, c'est-à-dire le film Avatar de James Cameron. On part donc du balisage standard en HTML4, sans aucune notion sémantique :

```
<div>
 <h1>Avatar</h1>
 <p>Réalisateur : James Cameron (born August 16, 1954)</p>
 <p>Science fiction</p>
 <p><a href="../movies/avatar-theatrical-trailer.html">Trailer</a></p>
</div>
```

On commence à préparer le document pour les microdonnées quand...

```
<div itemscope itemtype="...">
  <h1>Avatar</h1>
  <p>Réalisateur : James Cameron (born August 16, 1954)</p>
  <p>Science fiction</p>
  <p><a href="../movies/avatar-theatrical-trailer.html">Trailer</a></p>
</div>
```

Mais quel vocabulaire va-t-on utiliser pour décrire un film ? L'objectif ici étant d'utiliser le contenu de Schema.org, on va donc l'exploiter.

Depuis la page d'accueil (Lien 79), on trouve un lien vers les schémas (Lien 97), puis un autre vers la liste complète des types (Lien 98). Avant de se perdre dans cette longue liste, on peut observer un peu plus et remarquer qu'il y a un vocabulaire Movie dès la page des schémas : Lien 99. Suivant ce dernier lien, on tombe sur le vocabulaire recherché et sa documentation. On peut donc l'exploiter :

```
<div itemscope
itemtype="http://schema.org/Movie">
  <h1 itemprop="name"&g;Avatar</h1>
  <p itemprop="director" itemscope
itemtype="http://schema.org/Person">
        Réalisateur : <span itemprop="name">James Cameron</span> (born <span
itemprop="birthDate">August 16, 1954)</span>
  </p>
  <p itemprop="genre">Science fiction</p>
  <p><a href="../movies/avatar-theatrical-trailer.html" itemprop="trailer">Trailer</a></p>
</div>
```

On a directement utilisé le vocabulaire Person (Lien 100), car c'est le type attendu pour la propriété director, selon la documentation de Movie : Lien 99. Sur cette même page, on peut aussi remarquer la présence d'autres exemples.

Ce n'est pas parce qu'un type est attendu qu'il est requis ! On peut créer une arborescence à l'aide de vocabulaires, mais on peut aussi se limiter à un seul niveau s'il n'y a pas beaucoup d'informations à définir pour un élément. Si on n'avait que le nom du réalisateur, on aurait pu écrire le code suivant :

```
<div itemscope
itemtype="http://schema.org/Movie">
  <h1 itemprop="name"&g;Avatar</h1>
  <p>Réalisateur : <span itemprop="director">James Cameron</span></p>
  <p itemprop="genre">Science fiction</p>
  <p><a href="../movies/avatar-theatrical-trailer.html" itemprop="trailer">Trailer</a></p>
</div>
```

Si l'information n'est pas présente sur la page, il est inutile de la présenter sous forme cachée (que ce soit un caché ou une balise <meta>). L'idéal est de ne marquer que l'information visible au visiteur. (Il faut se rappeler que Google pénalise les sites qui présentent un contenu différent aux moteurs de recherche et aux visiteurs : Lien 101)

Une fois que l'on a intégré le principe des microdonnées,

utiliser les vocabulaires Schema.org n'est donc pas compliqué.

4. L'organisation des vocabulaires

Tous les vocabulaires héritent du même : Thing (Lien 102), le moins spécifique, celui qui propose le moins de propriétés, car elles sont partagées par tous les autres vocabulaires.

Ensuite, des types généraux en héritent : CreativeWork, Event, Intangible, Organization, Person, Place et Product. Chacune de ces « catégories » peut encore être plus spécifiée, à l'infini. On retrouve ici une belle application des principes de la programmation orientée objet.

Ce n'est pas parce qu'un type dérive d'un autre qu'il doit définir de nouveaux attributs : il permet d'être plus précis que son parent, c'est parfois la seule raison qui a poussé à créer un nouveau type, c'est d'ailleurs tout le principe des vocabulaires.

Il faut aussi remarquer que certains vocabulaires sont présentés avec un astérisque dans la liste : il s'agit de doublons, de vocabulaires qui peuvent être classés à deux endroits.

5. Présenter les données pour qu'elles soient compréhensibles par la machine

5.1. Dates et heures

Tous les types ne sont pas forcément facilement compréhensibles par la machine. L'exemple le plus courant est les dates, dont l'interprétation varie fortement entre les régions. Par exemple, 01/02/09 peut autant signifier le premier février de l'an 2009 que le neuf janvier 2001, en fonction du format utilisé. Il faut donc utiliser un format standardisé, pour que tout le monde comprenne la même chose des mêmes données.

Pour une date, on utilisera le format yyyy-mm-dd ; on peut aussi spécifier l'heure : yyyy-mm-ddThh:mm. On aura, par exemple, respectivement, pour l'origine de l'heure POSIX, 1970-01-01 et 1970-01-01T00:00.

5.2. Énumérations

Comme très souvent, certaines propriétés ne peuvent prendre que quelques valeurs prédéfinies : pour un produit, il n'y a que quelques disponibilités possibles (allant de *en stock* à *en précommande*), ces cas possibles sont présentés dans l'énumération ItemAvailability : Lien 103. Il est donc intéressant de préciser cela au moteur de recherche, étant donné qu'il ne peut pas deviner le contenu de la balise :

```
<div itemscope
itemtype="http://schema.org/Offer">
  <span itemprop="name">Microdonnées</span>
  <span itemprop="price">0,00 €</span>
  <link itemprop="availability"
href="http://schema.org/InStock"/> Disponible !
</div>
```

5.3. Métadonnées

Toutes les données ne peuvent pas être facilement déclarées, car il n'y a pas à proprement parler de texte pour ces données (par exemple, la durée d'une vidéo n'est pas toujours affichée). On utilise dans ce cas la balise <meta> :

```
<div itemscope
itemtype="http://schema.org/Offer">
  <span itemprop="name">Microdonnées</span>
  <meta itemprop="price" content="0,00 €" />
</div>
```

6. L'extension des vocabulaires

Un seul site ne peut pas couvrir tous les besoins : rien n'est proposé pour le moment pour les données génomiques, par exemple. Il faut donc une manière élégante de parer à ce problème : l'extension des schémas. On peut ainsi étendre facilement les schémas existants, en créer de nouveaux au besoin. Les applications comprenant les microdonnées Schema.org seront donc à même de comprendre au moins une partie de ces nouvelles données. Si un schéma devient suffisamment populaire, les moteurs de recherche pourraient commencer à l'exploiter.

6.1. Conventions de nommage

Les types et énumérations sont écrits en CamelCase.

Les propriétés sont écrites en camelCase. Les propriétés qui peuvent prendre plusieurs valeurs (comme les parents ou les critiques) sont au pluriel tandis que celles qui n'ont qu'une valeur restent au singulier (date de naissance ou de publication).

6.2. Extension

Pour étendre quelque chose, il suffit de prendre l'objet de base (classe, propriété ou énumération), un slash et le nom du dérivé : Book/Novel, musicGroupMember/leadVocalist, etc. On peut continuer à l'infini de cette manière.

L'avantage est qu'on peut toujours comprendre au moins une partie du contenu : on n'a pas défini ce qu'était un Novel, on sait cependant que c'est un livre.

Pour reprendre l'exemple précédent en étendant tout ce qu'il est possible d'étendre :

```
<div itemscope
itemtype="http://schema.org/Offer/Immaterial">
  <span itemprop="name">Microdonnées</span>
  <span itemprop="price/euros">0,00 €</span>
  <link itemprop="availability"
href="http://schema.org/InStock/ReadyForSending"/
> Disponible et prêt à l'envoi !
</div>
```

7. Références

- La spécification des microdonnées en HTML5 : Lien 104.
- Getting started with schema.org : Lien 105.
- Extension Mechanism : Lien 106.

Retrouvez l'article de Thibaut Cuvelier en ligne : Lien 107

C++ 2011 : le Draft final international validé à l'unanimité

La norme sera publiée avant la fin de l'année

Grande nouvelle pour les développeurs C++ : mercredi 10 août dernier, la version de la nouvelle norme a été approuvée à l'unanimité des votants. Ce qui jusqu'à présent était communément appelé Révision C++0x devient un standard officiel de l'ISO : C++11 !

Difficile de lister en quelques mots toutes les nouveautés introduites dans cette nouvelle version de la norme tant elles sont nombreuses, fondamentales et utiles. Mais nous ne résistons pas au plaisir de citer : les fonctions lambdas, les variadic template, la sémantique de mouvement, les rvalues reference, l'inférence de type, les listes d'initialisations, les délégations de constructeurs, l'intégration de beaucoup de bibliothèques Boost, un modèle multitâche, l'extension des unions, les énumérations dépoussiérées, etc...

Que pensez-vous de cette nouvelle norme ? Quels nouveautés vont faciliter vos développement ?

Et maintenant, vers quels directions le langage doit-il s'orienter ?

Commentez cette news de Jean-Marc Bourguet en ligne : Lien 108

GLC_lib 2.2.0 est sortie

La bibliothèque OpenGL pour Qt supporte maintenant les écrans multitouch

La version de 2.2.0 de GLC_lib est sortie.

Voici la liste des modifications et corrections :
- prise en charge des applications Multi OpenGL contextes ;
- prise en charge des écrans Multi-touch ;
- correction d'erreurs concernant les fichiers include lors de la compilation.

Pour Information, GLC_lib a été utilisé pour la création d'un logiciel de démonstration de l'utilisation de la maquette numérique liée aux exigences fonctionnelles (FDMU) au salon du Bourget 2011 sur le stand EADS : Lien 109

Téléchargez GLC_lib 2.2.0 : Lien 110

Commentez cette news de Laurent Ribon en ligne : Lien 111

QxOrm en version 1.1.7,

l'ORM pour Qt supporte désormais la suppression logique

QxOrm 1.1.7 vient de sortir.
Voici les nouveautés de la version 1.1.7 :
- suppression logique (*soft delete*) : pour plus de détails sur cette nouvelle fonctionnalité, rendez-vous sur la FAQ de QxOrm (*Comment utiliser le mécanisme de suppression logique ?*) ;
- ajout de fonctions dans le namespace '*qx::dao*' pour mettre à jour un élément en prenant en compte une requête SQL : *update_by_query, update_optimized_by_query, etc ;*
- correction d'un bogue avec le type *QVariant* (utilisé en tant que propriété d'une classe persistante) : il est à présent possible d'insérer la valeur *NULL* dans la base de données.

Pour plus d'informations sur la bibliothèque QxOrm : Lien 112

Commentez cette news de QxOrm en ligne : Lien 113

Sortie de PySide 1.0.5,

le binding Python de Qt est déjà disponible pour Maemo 5 « Fremantle »

PySide est sorti en version 1.0.5, apportant quelques modifications qualifiées de majeures :
- les widgets des fichiers *ui* sont maintenant exportés dans le widget racine (voir Lien 114 dans la mailing list) ;
- pyside-uic génère maintenant des barres de menus sans parent sous Mac OS X ;
- optimisation du système de signaux et de slots.

En plus de l'habituelle liste de bogues corrigés :

```
 892 Segfault when destructing QWidget and QApplication has event filter installed
 407 Crash while multiple inheriting with QObject and native python class
 939 Shiboken::importModule must verify if PyImport_ImportModule succeeds
 937 missing pid method in QProcess
 927 Segfault on QThread code.
 925 Segfault when passing a QScriptValue as QObject or when using .toVariant() on a QScriptValue
 905 QtGui.QHBoxLayout.setMargin function call is created by pyside-uic, but this is not available in the pyside bindings
 904 Repeatedly opening a QDialog with Qt.WA_DeleteOnClose set crashes PySide
 899 Segfault with 'QVariantList' Property.
 893 Shiboken leak reference in the parent control
 878 Shiboken may generate incompatible modules if a new class is added.
 938 QTemporaryFile JPEG problem
 934 A __getitem__ of QByteArray behaves strange
 929 pkg-config files do not know about Python version tags
 926 qmlRegisterType does not work with QObject
 924 Allow QScriptValue to be accessed via []
 921 Signals not automatically disconnected on object destruction
 920 Cannot use same slot for two signals
 919 Default arguments on QStyle methods not working
 915 QDeclarativeView.scene().addItem(x) make the x object invalid
 913 Widgets inside QTabWidget are not exported as members of the containing widget
 910 installEventFilter() increments reference count on target object
 907 pyside-uic adds MainWindow.setMenuBar(self.menubar) to the generated code under OS X
 903 eventFilter in ItemDelegate
 897 QObject.property() and QObject.setProperty() methods fails for user-defined properties
 896 QObject.staticMetaObject() is missing
 916 Missing info about when is possible to use keyword arguments in docs [was: QListWidgetItem's constructor ignores text parameter]
 890 Add signal connection example for valueChanged(int) on QSpinBox to the docs
 821 Mapping interface for QPixmapCache
 909 Deletion of QMainWindow/QApplication leads to segmentation fault
```

Cette version est d'ores et déjà disponible pour Maemo Fremantle, dans le dépôt extras-devel.

Sources : Lien 115 et Lien 116.
Téléchargez PySide 1.0.5. : Lien 117

Commentez cette news de Thibaut Cuvelier en ligne : Lien 118

Documentation Qt, découvrez les cinquante pages supplémentaires traduites par l'équipe Qt

Bonjour à tous,

L'équipe Qt de Developpez.com a de nouveau travaillé dur sur la traduction de la documentation Qt pour vous fournir cinquante pages supplémentaires.

La liste complète des pages traduites (Lien 119) vous donnera un aperçu du travail effectué par l'équipe. L'accent a été mis sur les classes de la vue graphique (QgraphicsScene : Lien 120, QgraphicsView : Lien 121, etc.), les classes de vue de données (QlistView : Lien 120, QtableView : Lien 123, QtreeView : Lien 124), les derniers événements qu'il nous restait à traduire (QtabletEvent : Lien 125, QsymbianEvent : Lien 126, etc.), ainsi que sur QpainterPath : Lien 127.

Vous retrouverez la liste des classes (Lien 128) et la vue d'ensemble (Lien 129) accompagnées de petits drapeaux français pour vous indiquer les pages traduites.

Un grand merci à toutes les personnes qui ont participé à ces traductions et relectures !

Nous sommes toujours à la recherche de personnes pour aider notamment sur les traductions de la documentation de Qt, n'hésitez pas à contacter un responsable Qt si vous êtes intéressé.

Commentez cette news de Jonathan Courtois en ligne : Lien 130

Qt 4.8 disponible en beta

avec Qt Quick 1.1 et Qt WebKit 2.2 et de nombreuses améliorations de performances

À peu près un mois après la sortie de la TP (Technology Preview), l'équipe de Qt vient de publier la première beta de Qt 4.8, grâce aux nombreux commentaires sur le blog anglophone.

Cette version ne devrait pas être considérée comme une version proche de la sortie, d'une RC et ne sera disponible qu'en mise à jour du Qt SDK 1.1 ou en sources (dans la catégorie Experimental de l'outil de maintenance). Symbian sera supporté comme plateforme expérimentale dans une version ultérieure de Qt 4.8. Les développeurs Qt pour Symbian doivent donc rester avec le Qt SDK en version 1.1 et Qt 4.7.3.

Au niveau du contenu, on peut remarquer que plusieurs sous-projets importants ont été mis à jour : notamment, Qt Quick 1.1 et Qt WebKit 2.2. Comme souvent, l'accent a été mis sur la qualité et les performances du framework, bien qu'il faille noter plusieurs nouveautés :
- Qt Platform Abstraction (alias Lighthouse) : Lien 131, une couche d'abstraction propre pour porter QtGui sur de nouveaux systèmes, qui deviendra la manière préférée pour Qt 5 ;
- le support d'OpenGL threadé, de même que l'accès HTTP : Lien 132 ;
- un accès au système de fichiers optimisé.

Qt Quick 1.1, quant à lui, apporte ces quelques nouveautés :
- support de l'écriture de droite à gauche ;
- cache des images amélioré ;
- améliorations de l'entrée de texte et support d'un clavier virtuel avec une séparation de l'écran.

Qt 4.8 beta est disponible sur les Qt Labs : Lien 133

Commentez cette news de Thibaut Cuvelier en ligne : Lien 134

Les derniers tutoriels et articles

Le C++0x dans Qt

Alors que beaucoup sont enthousiastes au sujet des technologies QML (Lien 135) et JavaScript, quelques-uns d'entre nous continuent de coder en C++. Le C++ est sur le point d'être mis à jour : le C++11 (anciennement connu sous le nom C++0x). Le projet final a été approuvé en mars dernier par le comité de normalisation du C++ (lire l'annonce sur Developpez : Lien 136) et la spécification finale devrait être publiée cet été. Si vous ne le savez pas encore, je vous invite à lire les pages spécialisées, telles que Wikipédia (Lien 137) ou la FAQ C++0x (Lien 138).

1. L'article original

Le site Qt Labs (Lien 139) permet aux développeurs de Qt de présenter les projets, plus ou moins avancés, sur lesquels ils travaillent.

Cet article est la traduction de l'article C++0x in Qt (Lien 140) d' Olivier Goffart paru dans Qt Labs.

Cet article est une traduction d'un des tutoriels écrits par **Nokia Corporation and/or its subsidiary(-ies)** inclus dans la documentation de Qt, en anglais. Les éventuels problèmes résultant d'une mauvaise traduction ne sont pas imputables à Nokia.

2. Introduction

L'un des objectifs de C++0x est de rendre le langage plus accessible, ce qui est relativement possible, puisque cette

nouvelle norme ne fait qu'ajouter plus de choses à apprendre, mais avec l'espoir que la partie « la plus utilisée » du C++ soit plus facile et plus intuitive.

La bonne nouvelle est que vous n'avez pas besoin d'attendre pour l'utiliser. Vous pouvez commencer dès aujourd'hui. Le compilateur que vous utilisez supporte probablement déjà une partie de C++0x : GCC (Lien 141) ou MSVC 2010 (Lien 142).

Bien que vous puissiez déjà utiliser C++0x avec les versions de Qt plus anciennes telles que Qt 4.7, Qt 4.8 viendra avec plus de support pour certaines nouvelles fonctionnalités du C++0x.

3. Les nouvelles macros

Certaines macros sont définies si le compilateur supporte les nouvelles fonctionnalités :

```
Q_COMPILER_RVALUE_REFS
Q_COMPILER_DECLTYPE
Q_COMPILER_VARIADIC_TEMPLATES
Q_COMPILER_AUTO_TYPE
Q_COMPILER_EXTERN_TEMPLATES
Q_COMPILER_DEFAULT_DELETE_MEMBERS
Q_COMPILER_CLASS_ENUM
Q_COMPILER_INITIALIZER_LISTS
Q_COMPILER_LAMBDA
Q_COMPILER_UNICODE_STRINGS
```

4. Les listes d'initialisation

Qt 4.8 ajoute de nouveaux constructeurs à Qvector (Lien 143), Qlist (Lien 144) et QstringList (Lien 145), qui vous permettent de les initialiser en utilisant des crochets d'initialisation. Vous pouvez maintenant faire :

```
QVector<int> testsData { 1, 2, 10, 42, 50,
123  };
QStringList options = { QLatin1String("foo"),
QLatin1String("bar")  };
```

qui initialise les conteneurs avec ces éléments.

5. Les références rvalue et la sémantique move

La plupart des classes Qt sont implicitement partagées (Lien 146), ce qui signifie qu'il est efficace de les copier si vous ne les modifiez pas (Copy-On-Write). Ce n'est pas le cas pour les std::vector (Lien 147), où chaque copie implique de copier toutes les données.

Donc, si vous avez un code comme ça :

```
std::vector<int> m_foo;
...
m_foo = getVector();
```

le getVector peut construire un nouveau std::vector et le retourner comme un objet temporaire. Ensuite, l'opérateur std::vector::operator= va supprimer l'ancien contenu de m_foo puis copier toutes les données à partir du vecteur temporaire dans m_foo. À la fin de l'instruction, le vecteur temporaire sera détruit et son destructeur va supprimer ses données. Ce serait plus efficace si, à la place, l'opérateur = permettait d'échanger des données de m_foo avec les données du vecteur temporaire. De cette façon, les anciennes données de m_foo seraient supprimées lorsque le vecteur temporaire est détruit et nous n'aurions pas besoin de créer une copie inutile. C'est ce qu'apporte la sémantique *move* dans C++11 et c'est obtenu grâce à des références rvalue.

Même si la copie d'une classe implicitement partagée est peu coûteuse, ça n'est pas totalement gratuit, nous devons encore augmenter et diminuer le compteur de références et les appels à l'opérateur = ne peuvent pas être inlinés, car cela accède aux données privées (nous ne pouvons pas inliner cela pour faciliter la compatibilité binaire). Alors maintenant, regardons l'opérateur = de Qimage (Lien 148) utilisant la sémantique *move* dans Qt 4.8 :

```
#ifdef Q_COMPILER_RVALUE_REFS
    inline QImage &operator=(QImage &&other)
    { qSwap(d, other.d); return *this; }
#endif
```

Nous venons d'échanger les données internes des deux images, c'est une opération très bon marché par rapport à l'opération normale qui nécessite un appel de fonction. Nous avons ajouté cette fonctionnalité à la plupart des classes implicitement partagées dans Qt. Comme l'opérateur = de nos conteneurs est beaucoup utilisé avec des temporaires, ça peut apporter de petites améliorations de performances de Qt, ce qui pourrait être une raison pour compiler Qt avec le support du C++0x (voir ci-dessous).

6. Les boucles for basées sur un range

Qt avait déjà un foreach très pratique (Lien 149), que vous pouvez aussi trouver dans d'autres bibliothèques, comme Boost (Lien 150). Le foreach de Qt fonctionne avec une macro compliquée, mais le C++0x va plus loin et intègre cette fonction dans le langage. Donc, au lieu d'écrire :

```
foreach(const QString &option, optionList)
{ ... }
```

vous pouvez écrire :

```
for (const QString &option : optionList) { ... }
```

Il y a une légère différence entre les deux : Qt fait une copie du conteneur avant de réaliser l'itération. C'est peu coûteux pour les conteneurs de Qt, puisqu'ils utilisent le partage implicite, mais elle l'est plus avec des conteneurs standards, qui réalisent une copie complète de tout le contenu. La version C++11 basée sur le *range* ne fait pas de copie, ce qui signifie que le comportement est indéfini si vous ajoutez ou supprimez des éléments dans le conteneur que vous parcourez. La nouvelle boucle for appellera les fonctions begin() (Lien 151) et end() (Lien 152), qui feront une copie du conteneur Qt s'il est partagé et non constant. Par conséquent, il est préférable de passer les conteneurs en constant.

```
template<class T> const T &const_(const T &t)
{ return t; }
for(auto it : const_(vector)) { ... }
```

7. Les lambdas

Nous avons testé les lambdas avec quelques-unes des fonctions de QtConcurrent : Lien 153. Nous les avons testées avec QtConcurrent::run (Lien 154) et QtConcurrent::map (Lien 155), mais cela ne fonctionne pas encore pour QtConcurrent::mapped (Lien 156), nous essaierons de résoudre ce problème à l'avenir. Ainsi, l'exemple de mise à l'échelle présenté dans la documentation de QtConcurrent::map pourrait être réécrit comme ceci :

```
QList<QImage> images = ...
QFuture<void> future = QtConcurrent::map(images,
[](QImage &image) {
          image = image.scaled(100,100);
      });
```

Je travaille également sur l'utilisation des lambdas dans les connexions signaux/slots, mais ce sera pour Qt5.

8. Les chaînes de caractères Unicode

Nous n'avons pas encore ajouté le support des nouveaux types de chaînes. Mais cela pourrait venir plus tard.

9. Allez-y et testez-le !

Si vous utilisez MSVC 2010 (Lien 157), il n'y a rien à faire, vous pouvez déjà utiliser certaines fonctionnalités telles que les lambdas ou les références rvalue. Si vous utilisez gcc (Lien 158), vous devez passer l'option -std=c++0x. Vous pouvez faire cela dans votre fichier projet qmake comme ceci :

```
QMAKE_CXXFLAGS += -std=c++0x
```

Si vous voulez compiler Qt en utilisant C++0x, vous pouvez faire :

```
CXXFLAGS="-std=c++0x" ./configure
```

Qt compilé avec le C++0x respecte la compatibilité binaire avec le bon vieux C++. Et vous n'avez pas besoin de compiler Qt avec C++0x pour écrire votre propre application en utilisant C++0x.

Retrouvez l'article d'Olivier Goffart traduit par Guillaume Belz en ligne : Lien 159

PyQt : signaux, slots et dispositions

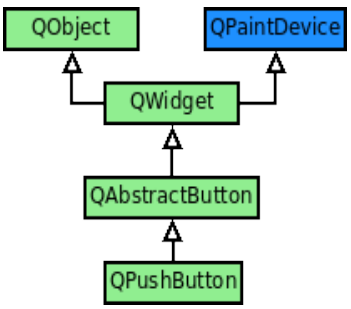
Après avoir vu ce à quoi une application PyQt simple ressemble au niveau du code, regardons de plus près l'interaction utilisateur. On va apprendre le modèle de connexion de signaux aux slots de Qt pour traiter les entrées et d'autres événements, ainsi que les dispositions (layouts) pour répartir de manière plus harmonieuse les widgets sur une fenêtre.

1. La hiérarchie des classes de PyQt

PyQt se base complètement sur les concepts orientés objet, il est donc important de comprendre comme les classes sont liées les unes aux autres.

Presque toutes les classes GUI dérivent de leur classe abstraite, qui définit un comportement commun pour des widgets similaires. Ces classes abstraites (ou toute classe de widget) hérite de QWidget, la classe de base de tous les composants GUI affichables à l'écran. QWidget hérite elle-même de QObject, une classe qui n'a rien à voir avec les GUI, mais forme la classe de base de la majorité des classes de PyQt et aide à fournir certaines fonctionnalités du framework.

Le diagramme de hiérarchie suivant montre cela très clairement pour la classe QPushButton :



La classe QPaintDevice aide à dessiner des choses à l'écran, elle est donc utilisée pour tout ce qui y est affichable.

Référence : Qwidget (Lien 160).

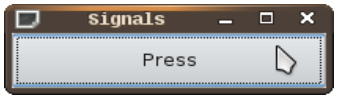
2. Signaux et slots

Toutes les interactions entre un utilisateur et un widget peuvent s'effectuer avec le concept de signaux et de slots. Quand un utilisateur effectue une action sur un widget (un clic, un appui de touche ou le pointeur qui survole), un événement est généré. Ces événements de base peuvent être gérés avec le concept de signaux et de slots.

Très simplement, un signal est quelque chose qui est généré à chaque fois qu'un événement a lieu. Le signal est émis en interne par les classes de PyQt ou les vôtres. Pour gérer ces signaux, on utilise des slots. On donne à chaque signal un slot auquel il est connecté. Une fois connecté à un slot, à chaque fois qu'il est émis, le signal est capturé par le slot et exécute une fonction prédéfinie pour gérer l'événement.

En d'autres mots, un signal est quelque chose qui est émis à chaque événement, un slot est quelque chose qui reçoit ce signal avec ses arguments et exécute une routine.

Pour exemplifier ce concept, on va utiliser un QPushButton avec un QWidget pour la fenêtre. Le code suivant lance une application avec un bouton qui, lorsqu'il est cliqué, ferme l'application. On le fait en connectant le signal connected() du bouton au slot quit() de l'application.




```
# Programme 02 - signaux
from PyQt4 import QtGui, QtCore
import sys

if __name__=='__main__':

    app = QtGui.QApplication(sys.argv)

    window = QtGui.QWidget()
    # On crée la fenêtre principale avec un
QWidget.
    window.setWindowTitle("Signaux")
    # On en définit le titre : "Signaux".
    button = QtGui.QPushButton("Appuyez", window)
    # On crée un bouton enfant, avec "Appuyez"
comme
        # texte, dans la fenêtre parente. En
spécifiant un
        # objet parent, ce nouveau widget y est
automatiquement
        # ajouté.
    button.resize(200, 40)
    # On redimensionne le bouton : (200, 40)
    button.connect(button,
QtCore.SIGNAL("clicked()"),\
        app, QtCore.SLOT("quit()"))
        # On connecte le signal clicked du bouton
au slot quit()
        # de QApplication
    window.show()
    # On affiche la fenêtre.

    app.exec_()
```

3. Connecter un signal à un slot

La connexion est effectuée pour que certains événements particuliers appellent leur slot respectif. Toutes les classes dérivant de QObject supportent ces connexions. La syntaxe de connect() est simple :

```
object.connect(Signal_Widget, Signal,
Slot_Widget, Slot)
```

La fonction connect() effectue une connexion entre deux ensembles de paramètres :

1. le signal :
   - il est constitué par l'objet qui émet le signal et le type de signal que l'on veut gérer (SIGNAL(QString)),
   - chaque widget émet un signal lors d'un événement, comme une interaction utilisateur ou une minuterie,
   - ces signaux ne doivent pas être confondus avec des signaux POSIX ou UNIX, c'est un concept très différent au niveau du système d'exploitation ;
2. le slot :
   - l'objet receveur et le slot qu'il doit exécuter dès la réception du signal (SLOT(QString)),
   - chaque signal émis peut être géré par une fonction, un slot,
   - un slot est exécuté à chaque fois qu'un signal qui lui est connecté est émis par la classe émettrice.

QtCore.QObject est la classe qui fournit cette fonction de connexion. Puisque la classe en dérive, on peut l'utiliser depuis son instance. On peut donc remplacer la vingtième ligne par :

```
QtCore.QObject.connect(button,
QtCore.SIGNAL("clicked()"),\
    app, QtCore.SLOT("quit()"))
```

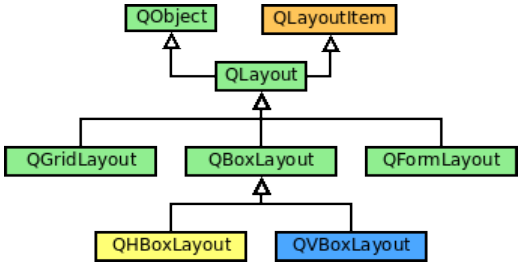
Un widget PyQt dispose de nombreux signaux et slots. Par exemple, un QAbstractButton possède les signaux clicked(), pressed(), released() et toggled(), ainsi que les slots click(), setChecked(Bool) et toggle(). PyQt fournit la majorité des fonctionnalités GUI que l'on peut imaginer.

Références : Qobject (Lien 161), signaux et slots (Lien 162), QpushButton (Lien 163) et Qapplication (Lien 164).

4. Dispositions (layouts)

En tentant de redimensionner l'exemple précédent, on a l'impression que le bouton est collé simplement dans la fenêtre. Il ne change pas de position, le contenu de la fenêtre ne change pas en fonction de l'écran de l'utilisateur. Les dispositions peuvent aider à ce niveau - et pour bien d'autres choses aussi.

Comme le mot l'indique, les dispositions aident à disposer les widgets sur une fenêtre de manière propre. QLayout est la forme la plus basique de disposition et possède quatre sous-classes principales, de type Box, Grid, Form et Stacked.



Utiliser une disposition est simple avec PyQt. On doit juste attacher la disposition voulue à une fenêtre ou un widget et on y ajoute des composants, qui seront automatiquement disposés. Le prochain exemple les utilise.

Références : Qlayout (Lien 165), classes de disposition (Lien 166)

5. Passer des arguments aux slots

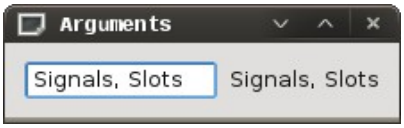
Les signaux et slots peuvent aussi transmettre des paramètres. Ceci est utile quand il y a une entrée ou quand des données doivent être envoyées automatiquement à chaque fois qu'un événement se produit.

Prenons un autre exemple, avec des paramètres. On va y connecter un éditeur sur une ligne à un label. Quand on entre du texte dans l'éditeur, des événements sont automatiquement générés et un slot du label va être exécuté pour changer le texte affiché pour celui qui vient

d'être tapé. Pour ceci, on utilise un QLabel pour le label et un QLineEdit pour l'éditeur.

Quelles sont les étapes à suivre pour obtenir le code ci-dessous ?

1. Importer les classes nécessaires.
2. Créer une instance de QApplication.
3. Créer une fenêtre (QWidget).
4. Y attacher un QHBoxLayout.
5. Créer un widget d'entrée et l'ajouter à la disposition.
6. Créer un widget d'affichage et l'ajouter à la disposition.
7. Connecter le signal textChanged(QString) de l'éditeur au slot setText(QString) du label.
8. Afficher la fenêtre.
9. Lancer la boucle principale.

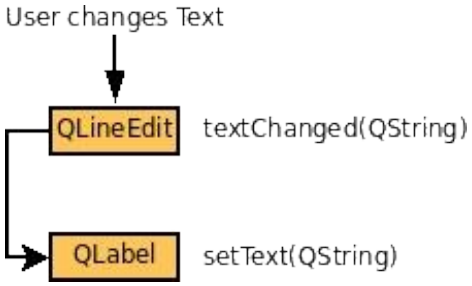



```
# Programme 03 - signaux et slots avec arguments
from PyQt4.QtCore import SIGNAL, SLOT
from PyQt4.QtGui import QApplication, QWidget,\
    QLineEdit, QLabel, QHBoxLayout
import sys

if __name__=='__main__':
    App = QApplication(sys.argv)
    Window = QWidget()
    Window.setWindowTitle("Arguments")
    Layout = QHBoxLayout(Window)
    Line = QLineEdit()
    Layout.addWidget(Line)
    Label = QLabel()
    Layout.addWidget(Label)
    Line.connect(Line,
SIGNAL("textChanged(QString)"),\
        Label, SLOT("setText(QString)"))
    Window.show()
    App.exec_()
```

Quand l'utilisateur entre ou modifie le texte du widget d'édition, un signal textChanged(QString) est émis avec la nouvelle chaîne en paramètre. On l'intercepte et on y connecte le slot (et fonction) setText(QString) du label. Ainsi, la chaîne modifiée voyage de l'éditeur au label et est affichée, le tout pour chaque changement, fût-ce un caractère.

La disposition possède une fonction intéressante, addWidget(), qui prend un widget et l'ajoute à la géométrie de la disposition. Cette méthode prend aussi un paramètre de facteur d'extension et un autre d'alignement. La disposition ajuste automatiquement les tailles des widgets, en se basant sur la taille de l'entrée dans cet exemple. Si on augmente la taille de la fenêtre, la disposition va s'adapter de même.

Si le concept de signaux et de slots semble encore confus, le graphique ci-dessous va l'éclairer un peu plus.



Références : QlineEdit (Lien 167), Qlabel (Lien 168), QHBoxLayout (Lien 169).

6. Exercice

Utilisez un QPushButton dans une fenêtre QWidget et changez son texte à chaque fois qu'il est cliqué. Aussi, ajoutez une disposition pour la fenêtre avec le bouton.

Chaque fois que le bouton est cliqué, son texte doit changer. Par exemple, si le texte du bouton est Hello!, après un clic il devient PyQt et un dernier le fait revenir à Hello!. Assurez-vous qu'il s'agrandit en même temps que la fenêtre.

7. Notes

On peut passer une fonction directement à connect() pour servir de slot. Elle ne doit pas forcément être un slot fourni par PyQt.

On peut aussi également écrire ses propres signaux et slots pour certains événements.

Tous les composants GUI n'ont pas une classe abstraite. Elle n'existe qu'en cas de widget ou d'item qui peut être étendu de plus d'une manière.

Retrouvez l'article de Harsh traduit par Thibaut Cuvelier en ligne : Lien 170

Création d'un système de chat en Pascal

Cet article a pour but de vous apprendre à créer un système de chat entièrement en Pascal. Il fait suite au Défi Pascal 2010 : Création d'un système de chat (Lien 171).

1. Introduction

Dans ce tutoriel, vous apprendrez à créer un système de chat en Pascal, permettant à plusieurs personnes de communiquer entre elles. Pour ce tutoriel, les bases de la Programmation Orientée Objet (POO) ainsi que la connaissance des threads sont nécessaires.

2. Présentation

2.1. Analyse du problème

Le but recherché est de pouvoir échanger du texte entre plusieurs ordinateurs. Pour cela, nous avons le choix entre deux protocoles de communication principaux, le TCP et l'UDP.

UDP pour User Datagram Protocol est un protocole de communication simple : il ne nécessite pas de procédure de connexion avant l'envoi des données, on dit qu'il fonctionne en mode "non connecté". Son principal atout est la rapidité de transmission, mais il ne garantit pas la réception du paquet.

TCP pour Transmission Control Protocol est un protocole de communication fiable : il fonctionne en mode connecté, c'est-à-dire que l'envoi de données doit être précédé d'une phase de connexion avec l'ordinateur distant. Ce protocole garantit donc la réception du message, mais il est bien plus lent que UDP. C'est celui que nous allons utiliser.

2.2. Architecture

L'architecture du chat peut prendre deux formes :

- un ordinateur joue le rôle de serveur, il reçoit et envoie tous les messages aux différents clients ;
- chaque ordinateur est à la fois serveur et client, c'est-à-dire que chaque ordinateur s'occupe d'envoyer à tous les clients chaque message.

Par souci de simplicité, nous utiliserons la première configuration.

2.3. Outils et composants

Nous allons maintenant choisir un composant qui gère le protocole TCP. J'ai choisi la bibliothèque de composants Indy 10, qui propose deux composants simples à utiliser, IdTCPServer et IdTCPClient.

3. Fonctionnement des composants TCP Indy

Du côté serveur comme du côté client, il est nécessaire d'utiliser un thread qui aura pour rôle de vérifier régulièrement la réception des messages.

Le composant IdTCPServer crée automatiquement un thread (un IdContext) propre à chaque client lors de sa connexion. Ce thread appelle l'évènement OnConnect, puis l'évènement OnExecute en boucle. Cet évènement OnExecute nous permettra de vérifier régulièrement la réception d'un message. Lors de la déconnexion, l'évènement OnDisconnect est appelé. Pour communiquer avec un client particulier, il suffit de récupérer son IdContext dans la liste des IdContext du composant.

Le composant IdTCPClient ne crée pas de thread, il sera donc nécessaire de le faire nous-mêmes. Pour commencer l'échange, il faut lancer la procédure de connexion, et si aucune exception n'est levée, nous pourrons continuer l'échange. Notre thread aura pour rôle d'appeler en boucle la méthode servant à la lecture des messages.

Comme les évènements sont appelés à partir du thread, il faut, si nécessaire, utiliser les méthodes de synchronisation pour accéder aux objets ou aux éléments de la VCL (sous Delphi) ou LCL (sous Lazarus).

4. Mise en place du chat

Pour faire communiquer les composants, nous utiliserons les méthodes ReadLn et WriteLn de la classe IOHandler, donc les messages seront de simples chaines de caractères.

4.1. Serveur

Nous allons créer un serveur basique. À la réception d'un message, le serveur le renverra à tous les clients. Nous allons donc créer une fiche avec seulement un composant IdTCPServer dessus.

Mettre la propriété Active du composant à true, et DefaultPort à 3333 (par exemple)

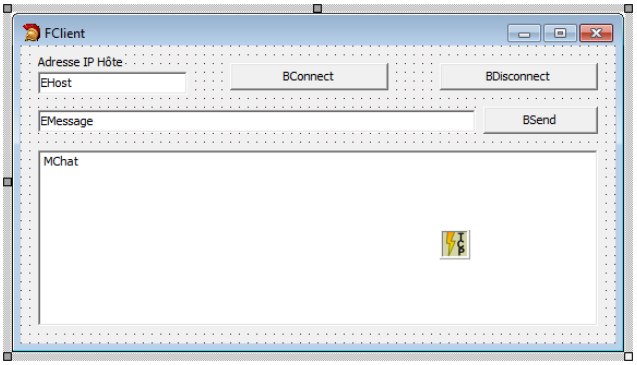
Les évènements OnConnection et OnExecute

```
procedure TFServeur.IdTCPServer1Connect(AContext:
TIdContext);
begin
  //Envoi d'un message au client qui vient de se
connecter
  AContext.Connection.IOHandler.WriteLn(' ---
Bienvenue sur le serveur --- ');
end;

procedure TFServeur.IdTCPServer1Execute(AContext:
TIdContext);
var Recept : string;
    ListeContext : TList;
    i : integer;
```

```
begin
  //Dès qu'un message est disponible, nous enregistrons l'IP du client et le message dans la variable
  Recept:=AContext.Binding.PeerIP+' : '+AContext.Connection.IOHandler.ReadLn;

  //Nous récupérons la liste des IdContext qui correspond à la liste des clients
  ListeContext:=IdTCPServer1.Contexts.LockList;

  //Pour chaque client, on envoie le message précédemment stocké dans la variable Recept
  for i:=0 to ListeContext.Count-1 do
    TIdContext(ListeContext.Items[i]).Connection.IOHandler.WriteLn(Recept);

  //Comme nous avons appelé LockList, nous devons débloquer la liste.
  IdTCPServer1.Contexts.UnlockList;

  //À partir d'ici, le composant va rappeler l'évènement OnExecute, après avoir vérifié que la connexion est toujours active
end;
```

4.2. Client

Pour le client, nous avons besoin d'une fiche un peu plus complète :



Exemple d'interface pour le client

Le champ Hôte permettra la saisie de l'adresse IP de l'ordinateur hôte. Notre fiche s'appelle FClient, et les composants sont :

Liste des composants

```
IdTCPClient1: TidTCPClient;  //Serveur TCP
EMessage: TEdit;             //Zone de saisie des messages
BConnect: TButton;           //Bouton de connexion
BDisconnect: TButton;        //Bouton de déconnexion
BSend: TButton;              //Bouton d'envoi des messages
MChat: TMemo;                //Zone d'affichage des messages
EHost: TLabeledEdit;         //Zone de saisie de l'adresse IP hôte
```

Comme nous avons dit plus haut, il nous faut créer un thread qui s'occupera de lire les nouveaux messages. Voici sa définition :

Définition du thread de réception

```
type
  //Le type de l'évènement OnReception
  TReceptEvent = procedure (Sender : TObject; Recept : string) of object;

  TReceptionthread = class(TThread)
  private
    FRecept : string;
    FIdClient : TIdTCPClient;
    FOnReception : TReceptEvent;
  protected
    procedure Execute; override;
    procedure DoOnReception;
  public
    constructor Create (CreateSuspended : boolean; AIdTCPClient : TIdTCPClient);

    //L'évènement OnReception
    property OnReception : TReceptEvent read FOnReception write FOnReception;
  end;
```

Le code associé à cet objet est :

Code du thread de réception

```
constructor TReceptionThread.Create
(CreateSuspended : boolean; AIdTCPClient : TIdTCPClient);
begin
  inherited Create(CreateSuspended);
  FIdClient:=AIdTCPClient;
end;

procedure TReceptionThread.DoOnReception;
begin
  //Si l'évènement OnReception a été assigné, alors on l'exécute en lui fournissant les bons paramètres
  if Assigned(FOnReception) then
    FOnReception(Self,FRecept);
end;

procedure TReceptionThread.Execute;
begin
  //On exécute en boucle jusqu'à ce que le thread se termine
  while not Terminated do
  begin
    FRecept:='';

    //On utilise une structure Try Except qui permet de détecter la déconnexion du client.
    //Si le client se déconnecte, ReadLn provoque une exception, et donc le thread se terminera.
    try
      FRecept:=FIdClient.IOHandler.ReadLn;
    except
      Terminate;
    end;

    if FRecept<>'' then
    //On synchronise l'évènement OnReception pour pouvoir utiliser les éléments de la VCL
      Synchronize(DoOnReception);
  end;
end;
```

Nous allons maintenant coder la fiche pour qu'elle réponde à nos attentes : après avoir cliqué sur Connecter,

l'utilisateur pourra taper du texte dans le champ associé et l'envoyer grâce au bouton Envoyer. Voici le code associé à la fiche, n'oubliez pas d'ajouter les définitions suivantes dans l'objet FClient.

Définition dans l'objet FClient

```
FReceptionThread : TReceptionThread;
procedure ReceptionMessage (Sender : TObject;
Recept : string);
```

Code de la fiche FClient

```
procedure TFClient.BConnectClick(Sender:
TObject);
begin
  //Nous vérifions si l'adresse IP saisie est
valide, dans le cas contraire, nous en informons
l'utilisateur
  if not GStack.IsIP(EHost.Text) then
    ShowMessage('L''adresse IP saisie est
invalide.')
  else
  begin
    try
      //Nous nous connectons. Si cette ligne
déclenche une exception, la suite ne sera pas
exécutée.
      //Le port saisi ici doit être le même que
celui du serveur.
      IdTCPClient1.Connect(EHost.Text,3333);

      //Nous créons notre thread destiné à gérer
la réception des messages.
      FReceptionThread:=TReceptionthread.Create(f
alse,IdTCPClient1);
      // Nous lui affectons aussi son évènement
OnReception.
      FReceptionThread.OnReception:=ReceptionMess
age;

      //Nous activons les composants de notre
fiche
      BDisconnect.Enabled:=true;
      BConnect.Enabled:=false;
      BSend.Enabled:=true;
      EMessage.Enabled:=true;
      MChat.Enabled:=true;
    except
      ShowMessage('Connexion avec le serveur
impossible.');
    end;
  end;
end;

procedure TFClient.BDisconnectClick(Sender:
TObject);
begin
  //procédure de déconnexion
  IdTCPClient1.Disconnect;

  //Désactivation des composants de la fiche
  BDisconnect.Enabled:=false;
  BConnect.Enabled:=true;
  BSend.Enabled:=false;
  EMessage.Enabled:=false;
  MChat.Enabled:=false;
end;

procedure TFClient.BSendClick(Sender: TObject);
begin
  //Envoi du texte de l'édit
  IdTCPClient1.IOHandler.WriteLn(EMessage.Text);
  EMessage.Clear;
end;

procedure TFClient.ReceptionMessage (Sender :
TObject; Recept : string);
begin
  //Évènement OnReception de notre objet
TReceptionThread
  //Si nous n'avions pas utilisé la méthode
Synchronize, l'utilisation du Mémo Mchat pourrait
poser des problèmes.

  MChat.Lines.Add(Recept);
end;
```

Nous venons de créer un chat basique qui permet d'échanger du texte entre plusieurs ordinateurs.

Télécharger les sources du Système de Chat v1 : Lien 172

5. Tests et utilisation

Il est maintenant temps de tester notre système. Pour cela, il ne nous est pas nécessaire d'avoir deux ordinateurs connectés en réseau : un seul suffit. Il vous suffit de choisir comme adresse hôte l'adresse « 127.0.0.1 ». C'est une adresse de bouclage (localhost) qui correspond à l'ordinateur local. Elle permet de simuler un réseau sur un seul ordinateur.

Pour déboguer le serveur, il vous suffit de l'ouvrir avec votre compilateur, et d'exécuter en même temps le client. De même, si vous voulez déboguer le client, ouvrez-le dans votre compilateur et exécutez le serveur à part.

Pour tester votre système sur un réseau local, il vous suffit d'exécuter un serveur sur un ordinateur, et d'exécuter les clients sur tous les autres ordinateurs. L'adresse IP hôte à saisir devra être celle du serveur.

Sous Windows, pour obtenir l'adresse IP de votre ordinateur, ouvrez une invite de commande et saisissez « ipconfig ». Vous n'avez plus qu'à lire l'adresse IPv4 de la carte réseau qui correspond à la connexion utilisée (Réseau sans fil si vous êtes connecté par Wi-fi, Réseau local sinon).
Sous Linux, l'adresse IP s'obtient en saisissant la commande « ifconfig » dans la console.

6. Amélioration du système et création d'un protocole de communication

Notre système fonctionne, mais il est très limité. Il se contente d'envoyer du texte entre plusieurs utilisateurs. Il serait intéressant, par exemple, de savoir qui est réellement connecté grâce à des pseudos, ainsi qu'une liste des personnes connectées. Nous devrons donc différencier les messages « texte » (envoyés par les utilisateurs, destinés à être affichés) des messages « techniques » (destinés à effectuer une action précise, par exemple, la connexion d'un nouveau client doit ajouter une ligne à la liste des personnes connectées).

Il apparaît un problème : nous devons être capables d'envoyer dans un seul et même message plusieurs informations. C'est pour cela que nous allons créer un

protocole de communication qui nous permettra de gérer tout cela facilement.

6.1. Le protocole de communication

Nous allons donc définir un protocole de communication personnel, qui nous permettra d'envoyer plus d'informations qu'un simple texte. Autrement dit, nous allons nous imposer une structure pour les chaines de caractères, qui nous permettra de stocker plusieurs informations à la fois, mais aussi de différencier les messages purement techniques d'avec les messages « texte ». De plus, pour éviter d'envoyer dans chaque message le pseudonyme de l'émetteur, nous allons leur attribuer un identifiant unique (ID) lors de leur connexion. Cet identifiant sera simplement un entier, qui sera envoyé sur trois caractères dans les messages.

Voici un exemple de protocole que nous pouvons utiliser :

Types de messages du serveur vers le client

Repère	Structure	Signification	Exemple
*	* + ID + Contenu	Message « texte »	'*001Salut'
O	O + ID	Connexion établie	'O001'
N	N + Raison	Connexion échouée	'NPseudo déjà utilisé'
C	C + ID + Pseudo	Annonce de connexion d'un membre	'C001Mick605'
D	D + ID	Annonce de déconnexion d'un membre	'D001'

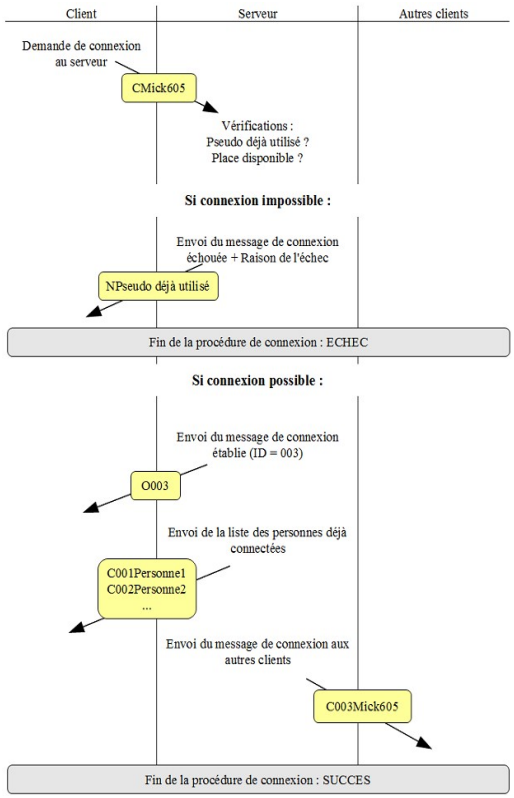
Types de messages du client vers le serveur

Repère	Structure	Signification	Exemple
*	* + ID + Contenu	Message « texte »	'*001Salut'
C	C + Pseudo	Demande de connexion au serveur	'CMick605'

Ainsi, chacune de ces structures de message sera analysée de manière différente, côté client aussi bien que côté serveur. Le premier caractère d'un message nous permet directement de différencier la procédure à suivre. Le découpage des messages et la récupération des informations seront simples, car nous savons que les ID sont composés de trois caractères.

6.2. Connexion d'un client

La connexion d'un client est un peu spéciale : le serveur doit vérifier plusieurs paramètres comme par exemple, si une place est disponible, ou que le pseudonyme n'est pas déjà utilisé par un autre client. De plus, le serveur doit fournir au client son identifiant ainsi que la liste des personnes connectées. Voici un schéma de la procédure de connexion d'un client.



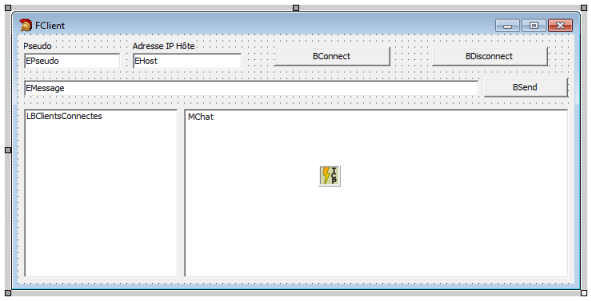
Chaque flèche représente l'envoi d'un message d'un ordinateur à l'autre. Dans chaque cadre jaune est noté un exemple de message pouvant être envoyé, en utilisant le protocole vu au chapitre 6.1.

6.3. Analyse et décomposition des messages

Maintenant que nous avons défini notre protocole, il nous faut créer les méthodes permettant de décomposer et d'analyser les chaines de caractères reçues, autant pour le serveur que pour le client.

6.3.1. Côté client

Nous modifions notre fiche pour y ajouter deux composants : un Edit nommé Epseudo pour saisir le pseudonyme, et une ListBox nommée LBClientsConnectes qui contiendra la liste des clients connectés.



Exemple d'interface pour le client

Nous modifions aussi notre code pour y inclure les variables nécessaires. Nous déclarons un entier nommé MonID destiné à contenir l'ID du client, et un tableau de chaine de caractères nommé Users qui contiendra le pseudonyme de tous les utilisateurs. L'identifiant ID jouera le rôle d'indice pour le tableau, c'est-à-dire que le client dont l'identifiant sera 003 sera stocké dans la case 3 du tableau Users.

Nous modifions le code de connexion :

Procédure de demande de connexion

```
procedure TFClient.BConnectClick(Sender:
TObject);
begin
  if EPseudo.Text='' then
    ShowMessage('Vous devez saisir un pseudo.')
  else
  begin
    //Nous vérifions si l'adresse IP saisie est
valide, dans le cas contraire, nous en informons
l'utilisateur
    if not GStack.IsIP(EHost.Text) then
      ShowMessage('L''adresse IP saisie est
invalide.')
    else
    begin
      try
        //Nous nous connectons. Si cette ligne
déclenche une exception, la suite ne sera pas
exécutée.
        //Le port saisi ici doit être le même que
celui du serveur.
        IdTCPClient1.Connect(EHost.Text,3333);

        //Envoi du message de demande de
connexion. À ce stade, nous ne sommes pas encore
en capacité de
        //communiquer avec les autres clients. Le
serveur doit d'abord vérifier quelques
paramètres, et
        //nous renvoyer une réponse
        IdTCPClient1.IOHandler.WriteLn('C'+EPseud
o.Text);

        //Nous créons notre thread destiné à
gérer la réception des messages.
        FReceptionThread:=TReceptionthread.Create
(false,IdTCPClient1);
        // Nous lui affectons aussi son évènement
OnReception.
        FReceptionThread.OnReception:=ReceptionMe
ssage;
      except
        ShowMessage('Connexion avec le serveur
impossible.');
      end;
    end;
  end;
end;
```

Nous devons aussi modifier le code permettant d'envoyer le texte (bouton BSend) pour prendre en compte le protocole que nous avons établi, c'est-à-dire que les messages texte doivent commencer par * suivi de l'ID de l'émetteur :

Procédure d'envoi d'un message Texte

```
procedure TFClient.BSendClick(Sender: TObject);
begin
  //Envoi du texte de l'edit
  IdTCPClient1.IOHandler.WriteLn('*'+Format('%3i'
,[MonID])+EMessage.Text);
  Emessage.Clear;
end;
```

Et nous modifions enfin notre procédure de réception des messages :

Procédure ReceptionMessage

```
procedure TFClient.ReceptionMessage (Sender :
TObject; Recept : string);
  //Évènement OnReception de notre objet
TReceptionThread


  function ExtractIDUser (Mess : string) :
integer;
  //fonction qui extrait les trois premiers
caractères du message et retourne l'ID
  begin
    if Length(Mess)>=2 then
      Result:=StrToIntDef(Mess[1]+Mess[2]+Mess[3]
,-1)
    else
      Result:=-1;
  end;


  procedure ConnectionSucceed (R : string);
  //R contient un message de connexion réussie de
la forme << ID >> qui signifie que nous sommes
connectés avec l'identifiant ID
  begin
    //MonID est une variable globale contenant
notre identifiant
    MonID:=ExtractIDUser(R);

    //Stockage de notre pseudo dans le tableau
d'utilisateurs
    Users[MonID]:=EPseudo.Text;

    //Nous annonçons notre connexion et nous
activons les composants nécessaires
    MChat.Lines.Add(' --- Connexion établie ---
');
    BDisconnect.Enabled:=true;
    BConnect.Enabled:=false;
    BSend.Enabled:=true;
    EMessage.Enabled:=true;
    MChat.Enabled:=true;
    EPseudo.Enabled:=false;
    LBClientsConnectes.Items.Add('- Liste des
clients connectés -');

    //Nous ajoutons notre pseudo dans la ListBox
    LBClientsConnectes.Items.Add(EPseudo.Text);
  end;

  procedure ConnectionFailed (R : string);
  //R contient un message de connexion échouée de
la forme << Raison >> qui signifie que nous ne
sommes pas connectés car : Raison
  begin
    //Nous affichons les informations nécessaires
    MChat.Lines.Add(' --- Connexion échouée ---
');
    MChat.Lines.Add('Raison : '+R);

    //Nous nous déconnectons du serveur
    IdTCPClient1.Disconnect;
  end;

  procedure UserConnection(R : string);
  //R contient un message de connexion de la
forme << ID + Pseudo >> qui signifie que Pseudo
vient de se connecter avec l'identifiant ID
  var ID : integer;
      Pse : string;
  begin
```

```
    //ID contient l'identifiant du message
    ID:=ExtractIDUser(R);
    //Pse contient le Pseudo
    Pse:=Copy(R,4,MaxInt);
    //Nous enregistrons l'utilisateur dans le
tableau
    Users[ID]:=Pse;

    //Nous annonçons la connexion du client
    MChat.Lines.Add(Pse+' vient de se
connecter.');

    //Nous ajoutons son nom dans la ListBox
    LBClientsConnectes.Items.Add(Pse);
  end;

  procedure UserDeconnection(R : string);
  //R contient un message de déconnexion de la
forme << ID >> qui signifie que la personne avec
l'identifiant ID vient de se déconnecter
  var ID : integer;
      Pse : string;
  begin
    //ID contient l'identifiant du message
    ID:=ExtractIDUser(R);
    //Pse contient le Pseudo sauvegardé lors de
la connexion
    Pse:=Users[ID];

    //Nous annonçons la déconnexion du client
    MChat.Lines.Add(Pse+' vient de se
déconnecter.');

    //Nous effaçons du tableau le client
déconnecté
    Users[ID]:='';

    //Nous cherchons et supprimons son nom dans
la ListBox
    I:=LBClientsConnectes.Items.IndexOf(Pse);
    if I<>-1 then
LBClientsConnectes.Items.Delete(I);
  end;

  procedure ReceptMessageTexte (R : string);
  //R contient un message texte de la forme << ID
+ Texte >> qui signifie que la personne avec
l'identifiant ID dit "Texte"
  var ID : integer;
      Pse : string;
      Texte : string;
  begin
    //ID contient l'identifiant du message
    ID:=ExtractIDUser(R);
    //Pse contient le Pseudo stocké lors de la
connexion
    Pse:=Users[ID];
    //Texte contient le texte envoyé par
l'utilisateur
    Texte:=Copy(R,4,MaxInt);

    //Nous affichons le texte précédé du pseudo
de l'émetteur
    MChat.Lines.Add(Pse+' dit :');
    MChat.Lines.Add(' > '+Texte);
  end;


var Repere : char;
begin
  //le premier caractère du message permet de
différencier la marche à suivre
  Repere:=Recept[1];
  //Nous l'effaçons de la chaine de caractères
  Delete(Recept,1,1);

  //Suivant les cas, on exécute une des
procédures ci-dessous
  case Repere of
    'O' : ConnectionSucceed(Recept);
    'N' : ConnectionFailed(Recept);
    'C' : UserConnection(Recept);
    'D' : UserDeconnection(Recept);
    '*' : ReceptMessageTexte(Recept);
  end;
end;
```

6.3.2. Côté serveur

Du côté du serveur, nous avons plusieurs éléments à ajouter. En effet, le serveur doit stocker la liste de tous les utilisateurs et leurs caractéristiques.

Premièrement, nous allons créer un type TUser, qui contiendra les caractéristiques d'un utilisateur (Pseudonyme, Identifiant, Statut de la connexion). Voici sa définition :

Définition des types TUser et TConnexionState

```
type
  TConnexionState = (csOk, csDisconnecting);

  TUser = class
  private
    FPseudo : string;
    FID : integer;
    FState : TConnexionState;
  public
    constructor Create (APseudo : string; AID :
integer);

    property Pseudo : string read FPseudo;
    property ID : integer read FID;
    property State : TConnexionState read FState
write FState;
  end;

...

constructor TUser.Create (APseudo : string; AID :
integer);
//Nous créons notre objet TUser en l'initialisant
avec les bonnes valeurs
begin
  inherited Create;

  FPseudo:=APseudo;
  FID:=AID;
  FState:=usOk;
end;
```

L'objet TIdContext qui, rappelons-le, correspond à une connexion avec un client, possède une propriété Data, de type Pointer, qui permet de pointer sur tout type d'information. Nous allons nous servir de cette propriété pour pointer sur un objet de type TUser. Ainsi, en possédant uniquement l'IdContext, nous pourrons retrouver les caractéristiques de l'utilisateur associé. Nous allons aussi stocker ces objets TUser dans un tableau

Users déclaré comme suit :

```
Users : array [1..MAXUSERS] of Tuser;
```

Nous définissons maintenant une procédure SendAll, qui aura pour but d'envoyer un message à tous les clients connectés.

Procédure SendAll

```
procedure TFServeur.SendAll (Mess : string);
var ListeContext : TList;
    i : integer;
    AUser : TUser;
begin
  //Nous récupérons la liste des IdContext qui correspond à la liste des clients
  ListeContext:=IdTCPServer1.Contexts.LockList;

  //Pour chaque client prêt (State = usOK), on envoie le message Mess
  for i:=0 to ListeContext.Count-1 do
  begin
    AUser:=TUser(TIdContext(ListeContext.Items[i]).Data);

    if Assigned(AUser) and (AUser.State=usOk) then
TIdContext(ListeContext.Items[i]).Connection.IOHandler.WriteLn(Mess);
  end;

  //Comme nous avons appelé LockList, nous devons débloquer la liste.
  IdTCPServer1.Contexts.UnlockList;
end;
```

Nous codons l'évènement OnDisconnect :

Événement OnDisconnect du IdTCPServer1

```
procedure TFServeur.IdTCPServer1Disconnect(AContext: TIdContext);
var AUser : TUser;
begin
  //On stocke dans une variable temporaire les caractéristiques de l'utilisateur qui se déconnecte
  AUser:=TUser(AContext.Data);

  if Assigned(AUser) then
  begin
    //On met son statut en Disconnecting pour éviter qu'il ne reçoive le message lors du SendAll suivant
    AUser.State:=csDisconnecting;

    //On envoie à tous les clients l'annonce de la déconnexion
    SendAll('D'+Format('%.3d',[AUser.ID])+AUser.Pseudo);

    //On efface le pointeur Data et on libère le client du tableau Users
    AContext.Data:=nil;
    Users[AUser.ID].Free;
  end;
end;
```

Et enfin, l'évènement OnExecute :

Évènement OnExecute du IdTCPServer1

```
procedure TFServeur.IdTCPServer1Execute(AContext: TIdContext);
var Recept : string;
begin
  //Lecture du message
  Recept:=AContext.Connection.IOHandler.ReadLn;

  //Analyse du message
  GestionMessage(AContext,Recept);
end;
```

La procédure GestionMessage permet d'analyser chaque message. Lors d'une demande de connexion, le serveur vérifie la place disponible, et si le pseudonyme est déjà utilisé. Lorsqu'il s'agit d'un message texte, le serveur se contente de le renvoyer à tous sans y effectuer de modifications.

Procédure de Gestion des messages

```
procedure TFServeur.GestionMessage (AContext : TIdContext; Recept : string);

  procedure EnvoiListeUsers;
  var i : integer;
  begin
    //On parcourt le tableau Users et on envoie la liste des personnes connectées à notre client
    for i:=1 to MAXUSERS do
      if Assigned(Users[i]) and (Users[i].State=csOk) then
        AContext.Connection.IOHandler.WriteLn('C'+Format('%.3d',[i])+Users[i].Pseudo);
  end;

  procedure DemandeConnection(R : string);
  //R contient un message de demande de connexion de la forme << Pseudo >> qui signifie que Pseudo veut se connecter
  var Raison : string;
      IsPossibleConnection : boolean;
      ID : integer;
  begin
    IsPossibleConnection:=true;

    //Nous testons si une personne ne possède pas le même pseudo
    ID:=0;
    repeat
      inc(ID);
    until (ID>MAXUSERS) or (Assigned(Users[ID]) and (Users[ID].Pseudo=R));

    if ID<=MAXUSERS then
    begin
      IsPossibleConnection:=false;
      Raison:='Pseudo déjà utilisé';
    end;

    if IsPossibleConnection then
    begin
      //Nous vérifions s'il reste de la place dans le serveur
      ID:=1;
      while (ID<=MAXUSERS) and Assigned(Users[ID]) do
        inc(ID);

      if ID>MAXUSERS then
      begin
```

```
        IsPossibleConnection:=false;
        Raison:='Serveur plein'
      end;
      //ici, la variable ID contient l'indice de
la première place libre dans le tableau Users
    end;

    //Maintenant que nos tests sont finis, nous
répondons à l'utilisateur
    if not IsPossibleConnection then
    begin
      //Connexion impossible
      AContext.Connection.IOHandler.WriteLn('N'+R
aison);
    end
    else
    begin
      //Connexion possible avec l'identifiant ID
      AContext.Connection.IOHandler.WriteLn('O'+F
ormat('%.3d',[ID]));

      //Envoi de la liste des personnes déjà
connectées
      EnvoiListeUsers;

      //Annonce de la connexion de l'utilisateur
à tous les autres utilisateurs
      SendAll('C'+Format('%.3d',[ID])+R);

      //Stockage dans le tableau Users
      Users[ID]:=TUser.Create(R,ID);

      //On fait pointer la propriété Data du
IdContext vers notre nouveau TUser
      AContext.Data:=Users[ID];
    end;

  end;
```

```
  procedure ReceptMessageTexte (R : string);
  //R contient un message texte de la forme << ID
+ Texte >> qui signifie que la personne avec
l'identifiant ID dit "Texte"
  begin
    SendAll('*'+R);
  end;


var Repere : char;
begin
  //Le premier caractère du message permet de
différencier la marche à suivre
  Repere:=Recept[1];
  //Nous l'effaçons de la chaine de caractères
  Delete(Recept,1,1);

  //Suivant les cas, on exécute une des
procédures ci-dessous
  case Repere of
    'C' : DemandeConnection(Recept);
    '*' : ReceptMessageTexte(Recept);
  end;
end;
```

Télécharger les sources du Système de Chat v2 : Lien 173

7. Conclusion

Nous possédons maintenant un système de chat fonctionnel, permettant de voir les utilisateurs connectés grâce à l'utilisation de pseudonymes. Ce système de chat peut servir de base pour prendre en charge plus de fonctionnalités, comme l'ajout de statuts, la personnalisation du texte, etc.

Retrouvez l'article de Mick605 en ligne : Lien 174

Application Perl/Tk non figée... les threads - Utilisation des méthodes internes et modules externes

Le but de cet article est d'expliquer comment empêcher une application Perl/Tk de se figer pendant une longue tâche.
Pour ce faire, nous parlerons des avantages et inconvénients des différentes méthodes : méthodes Tk, modules externes (threads, Win32...).

1. Introduction

La création de programmes Perl/Tk a généralement pour but d'éviter à son utilisateur d'avoir à travailler sous une console noire qui peut faire peur ! Le but est de pouvoir interagir avec l'application Perl via une interface graphique.

Nous sommes très souvent confrontés à des questions existentielles ! Supposons que notre application Perl/Tk ait pour rôle d'effectuer de longs calculs et que l'on souhaite afficher en même temps l'heure ou faire autre chose. Voici les différentes questions que l'on se pose au cours du développement :

- pourquoi ma fenêtre reste figée quand je clique sur le bouton ?
- pourquoi la fenêtre ne répond plus et qu'il y a une grosse tâche blanche ?
- comment dissocier mon calcul de ma fenêtre ?
- pourquoi l'heure ne bouge plus ?
- ...

Cet article va essayer de répondre à ces questions. Nous supposerons que vous avez déjà les bases de Perl et bien évidemment de Perl/Tk. Si ce n'est pas le cas, les cours de Perl et de Perl/Tk, sans oublier les FAQ, sont à votre disposition dans la rubrique Perl : Lien 175.

2. Problématique

Afin de vous exposer les problèmes rencontrés et expliquer comment les résoudre facilement, nous allons réaliser un script qui aura pour but de :

- rechercher les fichiers dans un répertoire au choix contenant notre motif de recherche ;
- afficher l'heure.

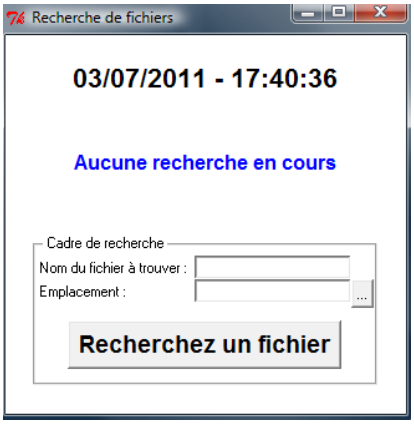
2.1. Tout-en-un

Pour résoudre notre problème, nous allons créer un programme que l'on nomme **recherche_fichier.pl** que voici :

recherche_fichier.pl

```
#!/usr/bin/perl
#================================================
# Auteur : djibril
# Date   : 03/07/2011 17:47:45
# But    : Script Perl/Tk utilisant pour
rechercher nos fichiers
#================================================
use Carp;
use strict;
use warnings;
use Tk;
use Tk::Dialog;
use Tk::LabFrame;
use File::Find;

my $fenetre = new MainWindow(
  -title      => 'Recherche de fichiers',
  -background => 'white',
);

# Affichage de l'heure
my $date_heure = date();
my $label_date = $fenetre->Label(
  -textvariable => \$date_heure,
  -background   => 'white',
  -font         => '{Arial} 16 {bold}',
)->pack(qw/ -pady 20 /);

# État de la recherche du fichier
my $etat_recherche = 'Aucune recherche en cours';
my $label_etat     = $fenetre->Label(
  -textvariable => \$etat_recherche,
  -background   => 'white',
  -foreground   => 'blue',
  -font         => '{Arial} 12 {bold}',
)->pack(qw/ -pady 20 /);

# Cadre de recherche
my $cadre = $fenetre->LabFrame(
  -label      => 'Cadre de recherche',
  -background => 'white',
)->pack(qw/ -pady 20 -padx 20 /);

my ( $motif_recherche, $repertoire_emplacement );
my $label1 = $cadre->Label( -text => 'Nom du
fichier à trouver : ', -background => 'white' );
my $entry_nom_fichier = $cadre->Entry(
-textvariable => \$motif_recherche );
my $label2 = $cadre->Label( -text => 'Emplacement
: ', -background => 'white' );
my $entry_emplacement = $cadre->Entry(
-textvariable => \$repertoire_emplacement );

my $bouton_emplacement = $cadre->Button(
  -text    => '...',
  -command => sub {
    $repertoire_emplacement = $cadre-
>chooseDirectory(
      -title     => 'Sélectionner un
emplacement',
      -mustexist => 1,
```

```
    );
  },
);

# Affichage d'un bouton pour rechercher un
fichier
my $bouton = $cadre->Button(
  -text    => 'Recherchez un fichier',
  -command => [ \&recherchez_fichier ],
  -font    => '{Arial} 14 {bold}',
);

$label1->grid( $entry_nom_fichier, '-',
-sticky => 'nw' );
$label2->grid( $entry_emplacement,
$bouton_emplacement, -sticky => 'nw' );
$bouton->grid( '-',                '-',
qw/ -padx 10 -pady 10 / );

# Centrer ma fenêtre
centrer_widget($fenetre);

# Toutes les secondes, la date et l'heure
évoluent
$fenetre->repeat( 1000, sub { $date_heure =
date(); } );

MainLoop;

#================================================
# But : Centrer un widget automatiquement
#================================================
sub centrer_widget {
  my ($widget) = @_;

  # Height and width of the screen
  my $largeur_ecran = $widget->screenwidth();
  my $hauteur_ecran = $widget->screenheight();

  # update le widget pour récupérer les vraies
dimensions
  $widget->update;
  my $largeur_widget = $widget->width;
  my $hauteur_widget = $widget->height;

  # On centre le widget en fonction de la taille
de l'écran
  my $nouvelle_largeur = int( ( $largeur_ecran -
$largeur_widget ) / 2 );
  my $nouvelle_hauteur = int( ( $hauteur_ecran -
$hauteur_widget ) / 2 );
  $widget->geometry( $largeur_widget . "x" .
$hauteur_widget . "+$nouvelle_largeur+
$nouvelle_hauteur" );

  $widget->update;

  return;
}

sub date {
  my $time = shift || time;     #$time par défaut
vaut le time actuel
  my ( $seconde, $minute, $heure, $jour, $mois,
$annee, $jour_semaine, $jour_annee,
$heure_hiver_ou_ete )
    = localtime($time);
  $mois  += 1;
  $annee += 1900;

  # On rajoute 0 si le chiffre est compris entre
1 et 9
  foreach ( $seconde, $minute, $heure, $jour,
$mois, $annee ) { s/^(\d)$/0$1/; }
  return "$jour/$mois/$annee - $heure:$minute:
$seconde";
}

sub recherchez_fichier {

  # Recherchons le fichier
  my $fichier_trouve = 0;
  $etat_recherche = 'Recherche de fichier en
cours...';
  find(
    { wanted => sub {
        if ( $_ =~ m{$motif_recherche}i )
{ $fichier_trouve++; }
      }
    },
    $repertoire_emplacement
  );
  $etat_recherche = "fichier trouvé :
$fichier_trouve fois";
  return;
}
```



Comme vous pouvez le constater, à l'exécution du programme, l'heure reste figée tant que la recherche de fichiers est en cours. Pour effectuer la recherche de fichiers, nous ne faisons pas appel à un programme externe, mais à une procédure que nous avons conçue « **recherchez_fichier** ». Pour éviter que notre programme ne reste figé, il est possible d'utiliser une méthode interne à Tk qui nous permettra de rafraîchir régulièrement la fenêtre afin de pouvoir voir l'heure défiler. La solution est d'utiliser la méthode **update**.

Exemple du fichier **recherche_fichier_update**. Nous avons juste effectué une légère modification de la procédure « recherchez_fichier » en faisant appel de la méthode **update** à chaque recherche de fichier. Cela permet de rafraîchir la fenêtre très régulièrement.

recherche_fichier_update.pl

```
sub recherchez_fichier {

  # Recherchons le fichier
  my $fichier_trouve = 0;
  $etat_recherche = 'Recherche de fichier en
cours...';
  $fenetre->update;
  find(
    { wanted => sub {
```

```
        if ( $_ =~ m{$fichier_recherche}i )
{ $fichier_trouve++; }
        $fenetre->update;
        }
    },
    $repertoire_emplacement
  );
  $etat_recherche = "fichier trouvé :
$fichier_trouve fois";
  $fenetre->update;

  return;
}
```

L'usage de la méthode « **update** » permet la plupart du temps de résoudre les soucis de fenêtres figées. Bien que, dans le cas précédent, l'affichage de l'heure ne soit pas strictement régulier, le comportement reste correct. Tout cela est possible car la tâche à exécuter est conçue au sein même de notre programme. Si nous devions faire appel à un programme externe, il serait impossible de mettre un « **update** » dans le programme, vu que nous n'aurions pas accès à son contenu.

2.2. Appel de programmes externes

Nous allons maintenant concevoir un programme Tk qui fait appel à un autre programme Perl, dont le but est de rechercher des fichiers en fonction d'un motif donné. Étant donné que nous parlons le langage Perl, ce programme sera traduit dans cette même langue !
Voici notre programme externe que l'on nomme « **trouve_fichier.pl** ». Il prend en arguments un motif et le nom d'un répertoire dans lequel la recherche s'effectue. Le résultat de la recherche est imprimé dans un fichier résultat dont le nom sera fourni en argument. Nous appellerons notre programme de la sorte :

```
wperl trouve_fichier.p -d "C:\repertoire" -m
"motif" -o "resultat.txt"
```

wperl permet de lancer Perl sans afficher la console DOS. Voici le code :

Programme externe : trouve_fichier.pl

```
#!/usr/bin/perl
#================================================
# Auteur : djibril
# Date   : 03/07/2011 18:10:19
# But    : Trouver les fichiers d'un répertoire
matchant avec un motif
#================================================
use Carp;
use strict;
use warnings;
use Getopt::Long;
use File::Find;

my ( $motif, $repertoire, $resultat_fichier ) =
();
GetOptions(
  'motif|m=s'      => \$motif,
  'repertoire|d=s' => \$repertoire,
  'output|o=s'     => \$resultat_fichier,
);

if ( ( not defined $motif ) || ( not defined
$repertoire ) || ( not defined
$resultat_fichier ) ) {
  die "USAGE : perl $0 -m <motif> -d <repertoire>
-o <resultat.txt>";
}

open my $fh, '>', $resultat_fichier
  or die("Impossible d'écrire dans le fichier
$resultat_fichier\n");

print {$fh} "Voici la liste de fichiers trouvés :
\n";

# Rechercher le fichier
find(
  { wanted => sub {
      if ( $_ =~ m{$motif}i ) {
        print {$fh} "- $File::Find::name\n";
      }
      }
  },
  $repertoire
);

close $fh;
```

Voici maintenant le programme Tk « **recherche_tk.pl** » appelant le programme externe :

recherche_tk.pl

```
#!/usr/bin/perl
use Carp;
use strict;
use warnings;
use Tk;
use Tk::LabFrame;

my $fenetre = new MainWindow(
  -title      => 'Recherche de fichiers',
  -background => 'white',
);

# Affichage de l'heure
my $date_heure = date();
my $label_date = $fenetre->Label(
  -textvariable => \$date_heure,
  -background   => 'white',
  -font         => '{Arial} 16 {bold}',
)->pack(qw/ -pady 20 /);

# État de la recherche du fichier
my $etat_recherche = 'Aucune recherche en cours';
my $label_etat     = $fenetre->Label(
  -textvariable => \$etat_recherche,
  -background   => 'white',
  -foreground   => 'blue',
  -font         => '{Arial} 12 {bold}',
)->pack(qw/ -pady 20 /);

# Cadre de recherche
my $cadre = $fenetre->LabFrame(
  -label      => 'Cadre de recherche',
  -background => 'white',
)->pack(qw/ -pady 20 -padx 20 /);

my ( $fichier_recherche,
$repertoire_emplacement );
my $label1 = $cadre->Label( -text => 'Nom du
fichier à trouver : ', -background => 'white' );
my $entry_nom_fichier = $cadre->Entry(
```

```
-textvariable => \$fichier_recherche );
my $label2 = $cadre->Label( -text => 'Emplacement
: ', -background => 'white' );
my $entry_emplacement = $cadre->Entry(
-textvariable => \$repertoire_emplacement );

my $bouton_emplacement = $cadre->Button(
  -text    => '...',
  -command => sub {
    $repertoire_emplacement = $cadre-
>chooseDirectory(
      -title     => 'Sélectionner un
emplacement',
      -mustexist => 1,
    );
  },
);

# Affichage d'un bouton pour rechercher un
fichier
my $bouton = $cadre->Button(
  -text    => 'Recherchez un fichier',
  -command => [ \&recherchez_fichier ],
  -font    => '{Arial} 14 {bold}',
);

$label1->grid( $entry_nom_fichier, '-',
-sticky => 'nw' );
$label2->grid( $entry_emplacement,
$bouton_emplacement, -sticky => 'nw' );
$bouton->grid( '-',                '-',
qw/ -padx 10 -pady 10 / );

# Centrer ma fenêtre
centrer_widget($fenetre);

# Toutes les secondes, la date et l'heure
évoluent
$fenetre->repeat( 1000, sub { $date_heure =
date(); } );

MainLoop;

#================================================
# But : Centrer un widget automatiquement
#================================================
sub centrer_widget {
  my ($widget) = @_;

  # Height and width of the screen
  my $largeur_ecran = $widget->screenwidth();
  my $hauteur_ecran = $widget->screenheight();

  # update le widget pour récupérer les vraies
dimensions
  $widget->update;
  my $largeur_widget = $widget->width;
  my $hauteur_widget = $widget->height;

  # On centre le widget en fonction de la taille
de l'écran
  my $nouvelle_largeur = int( ( $largeur_ecran -
$largeur_widget ) / 2 );
  my $nouvelle_hauteur = int( ( $hauteur_ecran -
$hauteur_widget ) / 2 );
  $widget->geometry( $largeur_widget . "x" .
$hauteur_widget . "+$nouvelle_largeur+
$nouvelle_hauteur" );

  $widget->update;
  return;
}

sub date {
  my $time = shift || time;    #$time par defaut
vaut le time actuel
  my ( $seconde, $minute, $heure, $jour, $mois,
$annee, $jour_semaine, $jour_annee,
$heure_hiver_ou_ete )
    = localtime($time);
  $mois  += 1;
  $annee += 1900;

  # On rajoute 0 si le chiffre est compris entre
1 et 9
  foreach ( $seconde, $minute, $heure, $jour,
$mois, $annee ) { s/^(\d)$/0$1/; }
  return "$jour/$mois/$annee - $heure:$minute:
$seconde";
}

sub recherchez_fichier {

  # Recherchons le fichier
  my $fichier_trouve = -1;
  $etat_recherche = 'Recherche de fichier en
cours...';
  $fenetre->update;

  # Lancement de la recherche
  system("trouve_fichier.pl -m $fichier_recherche
-d $repertoire_emplacement -o resultat.txt");

  # Lecture du fichier résultat
  open my $fh, '<', 'resultat.txt';
  while (<$fh>) { chomp; $fichier_trouve++; }
  close $fh;
  $etat_recherche = "fichier trouvé :
$fichier_trouve fois";
  $fenetre->update;

  return;
}
```

Comme vous pouvez le constater à l'exécution du programme, la fenêtre est bien figée pendant la recherche et il est impossible de faire autrement. Nous essayerons de trouver des solutions via des modules Perl.

3. Utilisation de modules Perl externes

Il existe des modules internes (threads (Lien 176) ...) et externes (Win32::Process (Lien 177) ...) nous permettant de jouer avec des processus. Nous allons en étudier quelques-uns.

3.1. Module Win32::Process

Comme vous l'aurez sans doute compris de part son nom, ce module n'est utilisable que sous Windows et vous devez l'installer. Ce module permet l'accès aux fonctions de contrôle de processus de l'API Win32. Il nous est ainsi possible de créer un nouveau processus dans lequel nous lancerons notre programme externe de recherche de fichiers. Ainsi, le programme externe tournera indépendamment de notre application Tk. Nous mettrons en place un moyen de vérifier que le processus tourne toujours ou est terminé afin de récupérer le résultat final. Avant de passer au programme en question, je vous

recommande de relire les notions de références en Perl, si vous n'avez pas l'habitude de les utiliser. Notre FAQ (Lien 178) pourra vous aider. Pour concevoir notre programme, nous modifierons la procédure « **recherchez_fichier** » et créerons une nouvelle « **verifier_resultat** ».

- **procédure « recherchez_fichier »**

Procédure recherchez_fichier
```
sub recherchez_fichier {

  # Recherchons le fichier
  $etat_recherche = 'Recherche de fichier en
cours...';
  $fenetre->update;

  # Lancement de la recherche
  my $fichier_resultat = 'resultat.txt';
  my $commande         = "trouve_fichier.pl
-m \"$motif_recherche\" "
    . " -d \"$repertoire_emplacement\"
-o \"$fichier_resultat\"";
  unlink $fichier_resultat;
  my $process_objet;
  Win32::Process::Create( $process_objet,
$EXECUTABLE_NAME, " $commande", 0,
NORMAL_PRIORITY_CLASS, '.' )
    || die
Win32::FormatMessage( Win32::GetLastError() );
  my $id;
  $id = $fenetre->repeat( 500, [
\&verifier_resultat, $process_objet,
$fichier_resultat, \$id ] );

  return;
}
```

Dans le code ci-dessus, nous remarquons que nous lançons le programme «**trouve_fichier.pl**» non pas à travers une commande système, mais via le module **Win32::Process**.

```
Win32::Process::Create( $process_objet,
$EXECUTABLE_NAME, " $commande", 0,
NORMAL_PRIORITY_CLASS, '.' )
    || die
Win32::FormatMessage( Win32::GetLastError() );
```

La méthode **Create** crée un nouveau processus.
Le premier argument est une variable qui contiendra l'objet du module.
Le deuxième contient le chemin vers l'exécutable perl (*perl.exe*). Nous avons utilisé la variable prédéfinie **$EXECUTABLE_NAME** de Perl issue du module appelé en début de script.

```
use English '-no_match_vars';
```

Le troisième argument contient la commande lancé par l'API Win32.
Le quatrième argument définit la priorité avec laquelle le processus sera lancée et le dernier précise le répertoire de travail du nouveau processus.

L'important est maintenant d'être capable de savoir quand le programme lancé sera terminé, car ce dernier n'est plus lié à notre application. Pour ce faire, nous allons appeler une procédure « **verifier_resultat** » via la fonction « **repeat** » de Perl/Tk qui permet de l'exécuter à un intervalle de temps en millisecondes.

```
my $id;
$id = $fenetre->repeat( 500, [
\&verifier_resultat, $process_objet,
$fichier_resultat, \$id ] );
```

La procédure est lancée toutes les 500 millisecondes et prend en argument l'objet du processus, le nom du fichier résultat (dans lequel sont listés tous les fichiers trouvés) et une référence à la variable $id. Cette dernière permettra de supprimer cet événement une fois que notre processus aura achevé sa tâche.

- **Procédure « verifier_resultat »**

Procédure verifier_resultat
```
sub verifier_resultat {
  my ( $process_objet, $fichier_resultat, $ref_id
) = @_;
  my $process_id = $process_objet-
>GetProcessID();
  my $exitcode;
  $process_objet->GetExitCode($exitcode);

  # Lecture du fichier résultat
  my $fichier_trouve = 0;
  open my $fh, '<', $fichier_resultat;
  while (<$fh>) { chomp; $fichier_trouve++; }
  close $fh;
  $fichier_trouve = $fichier_trouve > 0 ?
$fichier_trouve-- : 0;

  # Processus terminé
  if ( $exitcode == 0 ) {
    $etat_recherche = "Fini : $fichier_trouve
fichier(s) trouvé(s)";
    ${$ref_id}->cancel;
  }
  else {
    $etat_recherche
      = "Process : $process_id\n"
      . "Code de sortie : $exitcode\n"
      . "Fichiers en cours de recherche trouvés
$fichier_trouve fois";
  }

  return;
}
```

Dans notre procédure, nous récupérons l'id du processus via la méthode « **GetProcessID** ». Puis nous récupérons le code de sortie du processus. Ce code est **259** lorsque le processus est en cours. Nous lisons ensuite le fichier résultat afin de compter le nombre de fichiers trouvés. Si le nombre de fichiers est supérieur à 0, le nombre de fichiers est réduit de 1 car dans le fichier résultat, la première ligne est une ligne explicative et non un fichier trouvé.

Ensuite, nous vérifions le code de sortie du processus. Si ce dernier est « **0** », le processus est terminé et nous détruisons l'id (qui arrêtera le lancement la procédure « verifier_resultat ». Sinon, on met à jour l'affichage et la recherche continue.

```
Voici la liste de fichiers trouvés :
- C:/Article_Dvp/documents/perl-tk-threads-win32-poe/fichiers/perltk.odt
- C:/Article_Dvp/documents/perl-tk-threads-win32-poe/fichiers/recherche_fichier_update.pl
...
```

Remarque : Dans le gestionnaire de tâches, vous pouvez voir le processus lancé par l'API grâce au numéro du processus. Voici le programme final :

recherche_tk_win32 : programme final utilisant Win32::Process

```
#!/usr/bin/perl
use Carp;
use strict;
use warnings;
use Tk;
use Tk::LabFrame;
use Win32::Process;
use Win32;
use English '-no_match_vars';

my $fenetre = new MainWindow(
  -title      => 'Recherche de fichiers',
  -background => 'white',
);

# Affichage de l'heure
my $date_heure = date();
my $label_date = $fenetre->Label(
  -textvariable => \$date_heure,
  -background   => 'white',
  -font         => '{Arial} 16 {bold}',
)->pack(qw/ -pady 20 /);

# État de la recherche du fichier
my $etat_recherche = 'Aucune recherche en cours';
my $label_etat     = $fenetre->Label(
  -textvariable => \$etat_recherche,
  -background   => 'white',
  -foreground   => 'blue',
  -font         => '{Arial} 12 {bold}',
)->pack(qw/ -pady 20 /);

# Cadre de recherche
my $cadre = $fenetre->LabFrame(
  -label      => 'Cadre de recherche',
  -background => 'white',
)->pack(qw/ -pady 20 -padx 20 /);

my ( $motif_recherche, $repertoire_emplacement );
my $label1 = $cadre->Label( -text => 'Nom du
fichier à trouver : ', -background => 'white' );
my $entry_nom_fichier = $cadre->Entry(
-textvariable => \$motif_recherche );
my $label2 = $cadre->Label( -text => 'Emplacement
: ', -background => 'white' );
my $entry_emplacement = $cadre->Entry(
-textvariable => \$repertoire_emplacement );

my $bouton_emplacement = $cadre->Button(
  -text    => '...',
  -command => sub {
    $repertoire_emplacement = $cadre-
>chooseDirectory(
      -title     => 'Sélectionner un
emplacement',
      -mustexist => 1,
    );
  },
);

# Affichage d'un bouton pour rechercher un
fichier
my $bouton = $cadre->Button(
  -text    => 'Recherchez un fichier',
  -command => [ \&recherchez_fichier ],
  -font    => '{Arial} 14 {bold}',
);

$label1->grid( $entry_nom_fichier, '-',
-sticky => 'nw' );
$label2->grid( $entry_emplacement,
$bouton_emplacement, -sticky => 'nw' );
$bouton->grid( '-',                '-',
qw/ -padx 10 -pady 10 / );

# Centrer ma fenêtre
centrer_widget($fenetre);

# Toutes les secondes, la date et l'heure
évoluent
$fenetre->repeat( 1000, sub { $date_heure =
date(); } );

MainLoop;

#================================================
# But : Centrer un widget automatiquement
#================================================
sub centrer_widget {
  my ($widget) = @_;

  # Height and width of the screen
  my $largeur_ecran = $widget->screenwidth();
  my $hauteur_ecran = $widget->screenheight();

  # update le widget pour récupérer les vraies
dimensions
  $widget->update;
  my $largeur_widget = $widget->width;
  my $hauteur_widget = $widget->height;

  # On centre le widget en fonction de la taille
de l'écran
  my $nouvelle_largeur = int( ( $largeur_ecran -
$largeur_widget ) / 2 );
  my $nouvelle_hauteur = int( ( $hauteur_ecran -
$hauteur_widget ) / 2 );
  $widget->geometry( $largeur_widget . "x" .
$hauteur_widget . "+$nouvelle_largeur+
$nouvelle_hauteur" );

  $widget->update;

  return;
}

sub date {
  my $time = shift || time;    #$time par defaut
vaut le time actuel
  my ( $seconde, $minute, $heure, $jour, $mois,
$annee, $jour_semaine, $jour_annee,
$heure_hiver_ou_ete )
    = localtime($time);
  $mois  += 1;
  $annee += 1900;

  # On rajoute 0 si le chiffre est compris entre
```

```
1 et 9
  foreach ( $seconde, $minute, $heure, $jour,
$mois, $annee ) { s/^(\d)$/0$1/; }
  return "$jour/$mois/$annee - $heure:$minute:
$seconde";
}

sub recherchez_fichier {

  # Recherchons le fichier
  $etat_recherche = 'Recherche de fichier en
cours...';
  $fenetre->update;

  # Lancement de la recherche
  my $fichier_resultat = 'resultat.txt';
  my $commande          = "trouve_fichier.pl
-m \"$motif_recherche\" "
    . " -d \"$repertoire_emplacement\"
-o \"$fichier_resultat\"";
  unlink $fichier_resultat;
  my $process_objet;
  Win32::Process::Create( $process_objet,
$EXECUTABLE_NAME, " $commande", 0,
NORMAL_PRIORITY_CLASS, '.' )
    || die
Win32::FormatMessage( Win32::GetLastError() );
  my $id;
  $id = $fenetre->repeat( 500, [
\&verifier_resultat, $process_objet,
$fichier_resultat, \$id ] );

  return;
}

sub verifier_resultat {
  my ( $process_objet, $fichier_resultat, $ref_id
) = @_;
  my $process_id = $process_objet-
>GetProcessID();
  my $exitcode;
  $process_objet->GetExitCode($exitcode);

  # Lecture du fichier résultat
  my $fichier_trouve = 0;
  open my $fh, '<', $fichier_resultat;
  while (<$fh>) { chomp; $fichier_trouve++; }
  close $fh;
  $fichier_trouve = $fichier_trouve > 0 ?
$fichier_trouve-- : 0;

  # Processus terminé
  if ( $exitcode == 0 ) {
    $etat_recherche = "Fini : $fichier_trouve
fichier(s) trouvé(s)";
    ${$ref_id}->cancel;
  }
  else {
    $etat_recherche
      = "Process : $process_id\n"
      . "Code de sortie : $exitcode\n"
      . "Fichiers en cours de recherche trouvés
$fichier_trouve fois";
  }

  return;
}
```

Cette technique est très simple à mettre en place et vous permettra de pouvoir lancer des programmes externes effectuant des tâches très longues sans qu'ils ne bloquent votre application Perl/Tk. Étant donné que le processus tourne en tâche de fond, l'utilisateur de votre application peut continuer à travailler. Il peut même relancer x fois le même programme sans que les anciens ne soient terminés. C'est donc à vous de réfléchir à ce que vous souhaitez faire (par exemple, désactiver le bouton pendant que le processus tourne...). Vous pouvez tuer ce processus pour x raisons. Tout est possible avec un peu d'huile de coude !

Le seul inconvénient est qu'il permet de lancer certes, un programme externe, mais vous ne pouvez pas lancer une procédure Perl issue de votre programme. Normal, ce n'est pas son but. Pour ce faire, nous allons utiliser un autre module présent dans le Core de Perl : threads (Avec «**t**» en minuscule) (Lien 176).

3.2. Module threads

3.2.1. Qu'est ce qu'un thread ?

Un thread se traduit en français par processus léger. C'est un composant du processus principal (votre script). Chaque thread se partage la mémoire virtuelle du processus, mais possède sa propre pile d'appel (structure de données). Celui qui utilise des threads a donc l'impression que ces derniers travaillent en parallèle.

Il est important de ne pas confondre processus légers et multitâches, dont le principe est plutôt d'utiliser des processus différents.

3.2.2. Exemple basique d'utilisation

Pour la suite de cet article, vous devez installer les modules threads (Lien 176) et threads::shared (Lien 179). Ces modules font normalement déjà partie du CORE de Perl, mais la version présente n'est pas à jour et il se peut que certaines méthodes utilisées dans cet article ne fonctionnent pas. Je vous recommande donc de les installer.

Voici un script Perl (non Tk) qui vous permet de créer plusieurs threads. Il a pour but de créer dix threads qui afficheront deux lignes.

ExempleThreads
```
#!/usr/bin/perl
#=============================================
# Auteur : djibril
# But    : Exemple de threads
#=============================================
use warnings;
use strict;

use threads;

my @stockage_threads;

# Création de 10 threads.
for ( 0 .. 9 ) {
  $stockage_threads[$_] = threads->create( \&fun,
$_ );
}

print "Threads crées, passons à autre chose !\n";
sleep 2;
```

```
print "Allons récupérer nos valeurs de
retour...\n";

for ( 0 .. 9 ) {
  print "thread num $_  est terminé et nous
retourne la valeur : " . $stockage_threads[$_]-
>join() . "\n";
}

sub fun {
  my $number = shift;
  print "Bonjour, je suis le thread num :
$number\n";
  print "Mon id est : " . threads->tid() . "\n";
  sleep 2;
  print "le thread num $number meurt\n";

  return threads->tid();
}
```

Résultats

```
Threads créés, passons à autre chose!
Bonjour, je suis le thread num : 0
Mon id est : 1
Bonjour, je suis le thread num : 4
Mon id est : 5
Bonjour, je suis le thread num : 8
Bonjour, je suis le thread num : 3
Bonjour, je suis le thread num : 6
Bonjour, je suis le thread num : 1
Mon id est : 2
Bonjour, je suis le thread num : 2
Mon id est : 4
Bonjour, je suis le thread num : 5
Mon id est : 7
Bonjour, je suis le thread num : 7
Mon id est : 9
Bonjour, je suis le thread num : 9
Mon id est : 10
Mon id est : 6
Mon id est : 8
Mon id est : 3
Allons récupérer nos valeurs de retours...
le thread num 0 meurt
le thread num 4 meurt
thread num 0  est terminé et nous retourne la
valeur : 1
le thread num 1 meurt
thread num 1  est terminé et nous retourne la
valeur : 2
le thread num 6 meurt
le thread num 3 meurt
le thread num 8 meurt
le thread num 5 meurt
le thread num 9 meurt
le thread num 2 meurt
le thread num 7 meurt
thread num 2  est terminé et nous retourne la
valeur : 3
thread num 3  est terminé et nous retourne la
valeur : 4
thread num 4  est terminé et nous retourne la
valeur : 5
thread num 5  est terminé et nous retourne la
valeur : 6
thread num 6  est terminé et nous retourne la
valeur : 7
thread num 7  est terminé et nous retourne la
valeur : 8
thread num 8  est terminé et nous retourne la
valeur : 9
thread num 9  est terminé et nous retourne la
valeur : 10
```

Vous remarquez que l'on a créé dix threads qui se sont exécutés en même temps. C'est la raison pour laquelle les messages sont affichés dans un ordre aléatoire.
Le code ci-dessous nous permet de créer un thread et de le stocker (l'objet) dans un tableau.

```
$stockage_threads[$_] = threads->create( \&fun,
$_ );
```

La méthode **tid** nous retourne le numéro id du thread. La méthode **join** permet d'attendre que le thread se termine, de le nettoyer et de retourner les valeurs de la procédure lancée dans le thread (notamment &fun dans notre exemple). Si vous ne souhaitez pas récupérer la/les valeur(s) de retour de **join**, utilisez la méthode **detach** qui prend moins de ressources et détache votre script du thread. Celui-ci sera nettoyé proprement par Perl une fois qu'il sera terminé.

Il est très important de maîtriser un minimum les **threads** pour la suite de cet article.

Il est également impératif de comprendre comment partager des données entre le script Perl et ses threads ; c'est important pour la suite. Nous utilisons le module **threads::shared**.

Voici un exemple provenant du site enstimac (Lien 180), suivi des explications.

Partage de données - Code du site enstimac

```
use threads;
use threads::shared;

my $toto : shared = 1;
my $tata = 1;
threads->new(sub { $toto++; $tata++ })->join;

print "$toto\n"; # affiche 2 car $toto est
partagé
print "$tata\n"; # affiche 1 car $tata n'est pas
partagé
```

Dans le cas d'un tableau partagé, tous les éléments du tableau sont partagés, et pour une table de hachage partagée, toutes les clés et les valeurs sont partagées. Cela place des restrictions sur ce qui peut être affecté à des éléments de tableaux et de tables de hachage partagés : seules des valeurs simples ou des références à des variables partagées sont autorisées - de façon à ce qu'une variable privée ne puisse accidentellement devenir partagée. Une affectation incorrecte entraîne la mort du thread (die).
Par exemple :

Code du site enstimac

```
#!/usr/bin/perl
use threads;
use threads::shared;

my $var = 1;
my $svar : shared = 2;
```

```
my %hash : shared;

# ... créer quelques threads ...

$hash{a} = 1;       # pour tous les threads,
exists($hash{a}) et $hash{a} == 1
$hash{a} = $var;    # ok - copie par valeur : même
effet que précédemment
$hash{a} = $svar;   # ok - copie par valeur : même
effet que précédemment
$hash{a} = \$svar; # ok - référence à une
variable partagée
$hash{a} = \$var;   # entraîne la terminaison
(I<die>)
delete $hash{a};    # ok - pour tous les
threads, !exists($hash{a})
```

Le but de cet article n'est pas de vous faire un cours sur Perl et les threads, mais de vous exposer une méthode pour utiliser les threads avec Perl/Tk. Pour en savoir plus sur les threads, vous avez la documentation du module, des cours enstimac et une documentation interne à votre PC (**perldoc perlthrtut**).

Pour la suite de cet article, je considère que vous avez de bonnes notions sur les modules utilisés. De toute façon, les codes seront expliqués afin que vous puissiez les adapter à vos besoins.

3.2.3. Perl Tk et les threads

3.2.3.1. Avantages et inconvénients

Avantages

1. L'utilisateur peut continuer à interagir avec l'interface Perl/Tk pendant qu'une tâche s'effectue.
2. La fenêtre Perl/Tk n'est plus figée, car la tâche s'effectue dans un autre processus léger.
3. Il est possible de partager des données entre le script et les threads.

Inconvénients

1. La version actuelle de Perl/Tk (Tk-804.028) n'est pas ***"thread safe"*** d'après les auteurs.
2. L'utilisation des threads avec Perl/Tk n'est pas simple.
3. Le partage des données entre processus légers et/ou script principal n'est pas toujours évident.
4. Il est recommandé de créer ses threads avant tout code TK et ne pas faire apparaître de code TK dans les threads.
5. On ne peut donc pas créer des threads à la volée comme bon nous semble via un clic bouton.

Parmi les inconvénients de Perl/Tk et des threads, ayez surtout conscience des pièges même des threads.

Exemple :

- les threads peuvent modifier l'état du processus complet, affectant ainsi les autres threads ;
- **chdir** dans un thread modifie le répertoire courant des autres threads et du script principal (excepté sous Windows).

Lisez la documentation **BUGS AND LIMITATIONS** de la documentation CPAN du module **threads** et **threads::shared**.

Il est important de ne pas être surpris d'un mauvais comportement de votre script à cause d'une mauvaise maîtrise des modules **threads::*** !

3.2.3.2. Erreurs courantes

Pour vous montrer l'erreur classique que l'on est amené à faire la première fois que l'on souhaite utiliser les threads avec TK, on reprend notre exercice sur la recherche de fichiers.

thread_erreurs_classiques.pl

```
#!/usr/bin/perl
#================================================
# Auteur : djibril
# Date   : 03/07/2011 20:12:26
# But    : Script Perl/Tk utilisant pour les
threads - erreurs classiques
#================================================
use Carp;
use strict;
use warnings;
use Tk;
use Tk::LabFrame;
use File::Find;
use threads;

my $fenetre = new MainWindow(
  -title      => 'Recherche de fichiers',
  -background => 'white',
);

# Affichage de l'heure
my $date_heure = date();
my $label_date = $fenetre->Label(
  -textvariable => \$date_heure,
  -background   => 'white',
  -font         => '{Arial} 16 {bold}',
)->pack(qw/ -pady 20 /);

# État de la recherche du fichier
my $etat_recherche = 'Aucune recherche en cours';
my $label_etat     = $fenetre->Label(
  -textvariable => \$etat_recherche,
  -background   => 'white',
  -foreground   => 'blue',
  -font         => '{Arial} 12 {bold}',
)->pack(qw/ -pady 20 /);

# Cadre de recherche
my $cadre = $fenetre->LabFrame(
  -label      => 'Cadre de recherche',
  -background => 'white',
)->pack(qw/ -pady 20 -padx 20 /);

my ( $motif_recherche, $repertoire_emplacement );
my $label1 = $cadre->Label( -text => 'Nom du
fichier à trouver : ', -background => 'white' );
my $entry_nom_fichier = $cadre->Entry(
-textvariable => \$motif_recherche );
my $label2 = $cadre->Label( -text => 'Emplacement
: ', -background => 'white' );
my $entry_emplacement = $cadre->Entry(
-textvariable => \$repertoire_emplacement );

my $bouton_emplacement = $cadre->Button(
  -text    => '...',
  -command => sub {
```

```
    $repertoire_emplacement = $cadre->chooseDirectory(
      -title     => 'Sélectionner un emplacement',
      -mustexist => 1,
    );
  },
);

# Affichage d'un bouton pour rechercher un fichier
my $bouton = $cadre->Button(
  -text    => 'Recherchez un fichier',
  -command => sub {
    $etat_recherche = 'Recherche de fichier en cours...';
    $fenetre->update;
    my $Thread = threads->create( \&recherchez_fichier, $motif_recherche, $repertoire_emplacement );
    $Thread->detach();

    #$Thread->join();
    $etat_recherche = "fichier trouvé";
    $fenetre->update;
  },
  -font => '{Arial} 14 {bold}',
);

$label1->grid( $entry_nom_fichier, '-', -sticky => 'nw' );
$label2->grid( $entry_emplacement, $bouton_emplacement, -sticky => 'nw' );
$bouton->grid( '-',                '-', qw/ -padx 10 -pady 10 / );

# Centrer ma fenêtre
centrer_widget($fenetre);

# Toutes les secondes, la date et l'heure évoluent
$fenetre->repeat( 1000, sub { $date_heure = date(); } );

MainLoop;

#================================================
# But : Centrer un widget automatiquement
#================================================
sub centrer_widget {
  my ($widget) = @_;

  # Height and width of the screen
  my $largeur_ecran = $widget->screenwidth();
  my $hauteur_ecran = $widget->screenheight();

  # update le widget pour récupérer les vraies dimensions
  $widget->update;
  my $largeur_widget = $widget->width;
  my $hauteur_widget = $widget->height;

  # On centre le widget en fonction de la taille de l'écran
  my $nouvelle_largeur = int( ( $largeur_ecran - $largeur_widget ) / 2 );
  my $nouvelle_hauteur = int( ( $hauteur_ecran - $hauteur_widget ) / 2 );
  $widget->geometry( $largeur_widget . "x" . $hauteur_widget . "+$nouvelle_largeur+$nouvelle_hauteur" );
  $widget->update;

  return;
}

sub date {
  my $time = shift || time;    #$time par defaut vaut le time actuel
  my ( $seconde, $minute, $heure, $jour, $mois, $annee, $jour_semaine, $jour_annee, $heure_hiver_ou_ete )
    = localtime($time);
  $mois  += 1;
  $annee += 1900;

  # On rajoute 0 si le chiffre est compris entre 1 et 9
  foreach ( $seconde, $minute, $heure, $jour, $mois, $annee ) { s/^(\d)$/0$1/; }
  return "$jour/$mois/$annee - $heure:$minute:$seconde";
}

sub recherchez_fichier {
  my ( $motif_recherche, $repertoire_emplacement ) = @_;

  # Recherchons le fichier
  my $fichier_trouve = 0;
  find(
    { wanted => sub {
        if ( $_ =~ m{$motif_recherche}i ) {
          $fichier_trouve++;
          print "$fichier_trouve-$File::Find::name\n";
        }
        }
    },
    $repertoire_emplacement
  );

  return $fichier_trouve;
}
```

En exécutant ce programme, l'affichage est comme nous le souhaitons. Pendant l'affichage des fichiers trouvés, notre application n'est plus figée, mais on remarque ceci :

Utilisation de la méthode detach

1. la fenêtre Tk n'est pas figée ;
2. l'heure est actualisée régulièrement ;
3. le script s'arrête brusquement à la fin de l'exécution du thread avec un message de ce type *"Free to wrong pool 2ccee28 not 235e40 at C:/Perl/site/lib/Tk/Widget.pm line 98 during global destruction."*.

Utilisation de la méthode join à la place de detach

1. la fenêtre Tk reste figée tant que le thread n'est pas terminé (ce qui est bien dommage) ;
2. si on ajoute un **update** dans la procédure *"recherchez_fichier"*, on a un message d'erreur de ce type *Attempt to free non-existent shared string '_TK_RESULT_', Perl interpreter: 0x2ccc244 at ...* ;
3. Une fois le thread terminé, le script s'arrête

anormalement avec les mêmes messages d'erreur cités ci-dessus.

Pourquoi ces arrêts brusques du script ?

En fait, nous violons les règles actuelles de Perl/Tk car il n'est pas **"thread safe"**.
On ne doit absolument pas mettre de code Perl/Tk dans une procédure lancée dans un thread. Or, c'est le cas ici, puisque l'on fait un *update*.

Ne me demandez pas pourquoi et quelles sont ces règles ! Le *README* du module nous dit ceci : *Tk804.027 builds and loads into a threaded perl but is NOT yet thread safe.*

Les auteurs et personnes en charge de la maintenance de Perl/Tk ont prévu de le rendre *"thread-safe"* dans leur TODO (Lien 181) dans un futur proche ! En attendant ces nouveautés, on va utiliser un autre procédé qui est recommandé et plus sûr.

3.2.4. Mise en place des threads dans notre exemple

J'espère que vous n'êtes pas fatigué ! Après tous ces paragraphes et exemples de code, nous allons enfin voir comment créer proprement des threads en Perl/Tk !

Vous devez vous mettre en tête ceci :

1. on doit créer nos threads en début de script avant même d'écrire du code Tk ;
2. on ne doit pas faire appel à du code Perl Tk dans les procédures que l'on souhaite utiliser dans nos threads.

Je vais vous exposer le concept de notre script.

Nous allons créer un thread qui tournera en tâche de fond. Son but sera de dormir si on ne lui demande rien ou de travailler si on le met à contribution. Pour lui dire de travailler, on lui enverra un signal qu'il interceptera. Ce signal mentionne la procédure à appeler et on récupère le résultat de notre procédure.

Pour commencer, faisons appel aux modules dont on aura besoin.

Chargement des modules
```
#!/usr/bin/perl
#================================================
# Auteur : djibril
# Date   : 03/07/2011 14:08:50
# But    : Script Perl/Tk utilisant des threads pour rechercher nos fichiers
#================================================
use warnings;
use strict;

use Tk;                          # Pour créer notre GUI
use Tk::LabFrame;
use File::Find;
use threads;                     # Pour créer nos threads
use threads::shared;             # Pour partager nos données entre threads
use Time::HiRes qw( sleep );     # Pour faire des sleeps < à une seconde
```

Créons une liste associative dans laquelle on mentionne les fonctions à appeler dans notre thread. Ce hash a en clé le nom de la fonction et en valeur la référence à la procédure.

Liste associative contenant nos fonctions à lancer dans un thread
```
# Contient les fonctions à appeler dans le thread si besoin
my %fonctions_a_lancer_dans_thread = (
  'recherchez_fichier' => \&recherchez_fichier,
);
```

Ce hash sera visible dans notre thread car il a été déclaré avant même la création de ce dernier.

Déclarons maintenant les variables qui seront partagées entre le thread et le thread principal (le script) :

Déclaration des variables partagées
```
#==================================
# Threads et variables partagées
#==================================
my $tuer_thread : shared;          # Permet de tuer le thread proprement
my $nom_fonction : shared;         # Contient le nom de la fonction à appeler
my $thread_travail : shared;       # Contient la valeur permettant au thread de lancer une procédure
my @arguments_thread : shared;     # Contient les arguments à passer à une éventuelle procédure
my @resultat_fonction : shared;    # Contient le résultat des fonctions lancées dans le thread

$thread_travail = 0;               # 0 : thread ne fait rien, 1 : il bosse
$tuer_thread    = 0;               # 0 : thread en vie, 1 : thread se termine
```

Nous avons choisi de partager cinq variables :

1. **$tuer_thread** contient la valeur 0 ou 1. C'est ainsi que l'on demande au thread de mourir ou non ;
2. **$nom_fonction** contient le nom de la fonction que l'on souhaite appeler dans notre thread (grâce aux hash *%fonctions_a_lancer_dans_thread*) ;
3. **$thread_travail** contient la valeur 0 ou 1. C'est ainsi que l'on demande au thread de lancer une procédure ou de dormir ;
4. **@arguments_thread** contient les arguments que l'on souhaite passer aux procédures lancées dans le thread ;
5. **@resultat_fonction** contient les résultats de la procédure lancée dans le thread.

Ne vous inquiétez pas si cela reste ambigu pour l'instant, tout sera clarifié avec la suite du code !

Créons maintenant notre thread.

Création du processus léger (thread)
```
# Création du thread
my $thread = threads->create( \&notre_processus_leger );
```

Créons la procédure « **notre_processus_leger** » qui tournera continuellement dans le processus léger.

Processus léger : notre_processus_leger

```
#================================================
# notre_processus_leger
#================================================
sub notre_processus_leger {

  # Tourne en rond
  while (1) {

    # demande au thread de travailler
    if ( $thread_travail == 1 ) {

      # Lance la procédure
      $fonctions_a_lancer_dans_thread{$nom_fonction}->(@arguments_thread);

      # demande au thread de dormir
      $thread_travail = 0;
    }

    # Terminer le thread
    last if ( $tuer_thread == 1 );
    sleep 0.5;
  }
  return;
}
```

Explication :

Dans notre procédure, nous avons fait une boucle *while* infinie qui permet au thread de ne jamais mourir, sauf si on le lui demande. Dans un premier temps, le thread vérifie si la variable **$thread_travail** est à 1. Si c'est le cas, cela signifie que l'on a demandé au thread de lancer une procédure (on verra plus tard comment on s'y prend). Dans le cas contraire, on vérifie si le thread doit mourir ou dormir pendant une demi-seconde.

Maintenant que nos variables sont déclarées et partagées, et notre thread créé, passons au code Perl/Tk.

Code Perl/Tk

```
#==================================
# Début du code principal Perl Tk
#==================================
my $fenetre = new MainWindow(
  -title      => 'Recherche de fichiers',
  -background => 'white',
);

# Affichage de l'heure
my $date_heure = date();
my $label_date = $fenetre->Label(
  -textvariable => \$date_heure,
  -background   => 'white',
  -font         => '{Arial} 16 {bold}',
)->pack(qw/ -pady 20 /);

# État de la recherche du fichier
my $etat_recherche = 'Aucune recherche en cours';
my $label_etat     = $fenetre->Label(
  -textvariable => \$etat_recherche,
  -background   => 'white',
  -foreground   => 'blue',
  -font         => '{Arial} 12 {bold}',
)->pack(qw/ -pady 20 /);

# Cadre de recherche
my $cadre = $fenetre->LabFrame(
  -label      => 'Cadre de recherche',
  -background => 'white',
)->pack(qw/ -pady 20 -padx 20 /);

my ( $motif_recherche, $repertoire_emplacement );
my $label1 = $cadre->Label( -text => 'Nom du fichier à trouver : ', -background => 'white' );
my $entry_nom_fichier = $cadre->Entry( -textvariable => \$motif_recherche );
my $label2 = $cadre->Label( -text => 'Emplacement : ', -background => 'white' );
my $entry_emplacement = $cadre->Entry( -textvariable => \$repertoire_emplacement );

my $bouton_emplacement = $cadre->Button(
  -text    => '...',
  -command => sub {
    $repertoire_emplacement = $cadre->chooseDirectory(
      -title     => 'Sélectionner un emplacement',
      -mustexist => 1,
    );
  },
);

# Affichage d'un bouton pour rechercher un fichier
my $bouton = $cadre->Button(
  -text    => 'Recherchez un fichier',
  -command => [ \&recherchez_fichier_tk ],
  -font    => '{Arial} 14 {bold}',
);

$label1->grid( $entry_nom_fichier, '-', -sticky => 'nw' );
$label2->grid( $entry_emplacement, $bouton_emplacement, -sticky => 'nw' );
$bouton->grid( '-',                '-', qw/ -padx 10 -pady 10 / );

# Centrer ma fenêtre
centrer_widget($fenetre);

# Toutes les secondes, la date et l'heure évoluent
$fenetre->repeat( 1000, sub { $date_heure = date(); } );

MainLoop;
```

Notre fenêtre Tk affiche régulièrement l'heure grâce à la méthode Tk **repeat**. Elle contient des cadres et champs permettant de réceptionner le motif de recherche et le répertoire où chercher nos fichiers. Notre bouton « recherchez un fichier » lancera le nécessaire pour la recherche. À la fermeture de l'application, la procédure « fermer_application » est appelée et nous permet de fermer proprement notre programme et d'arrêter le thread lancé.

Fermeture application

```
sub fermer_application {

  # Demande au thread de se terminer
  $tuer_thread = 1;

  # On attend que le thread se termine proprement
  $thread->detach();
```

```
  exit;
}
```

Nous avons également les procédures « date » et « centrer_widget » permettant respectivement de donner la date et l'heure, puis de centrer un widget (notamment notre fenêtre).

```
#================================================
# But : Obtenir la date et l'heure
#================================================
sub date {
  my $time = shift || time;    #$time par defaut vaut le time actuel
  my ( $seconde, $minute, $heure, $jour, $mois, $annee, $jour_semaine, $jour_annee, $heure_hiver_ou_ete )
    = localtime($time);
  $mois  += 1;
  $annee += 1900;

  # On rajoute 0 si le chiffre est compris entre 1 et 9
  foreach ( $seconde, $minute, $heure, $jour, $mois, $annee ) { s/^(\d)$/0$1/; }
  return "$jour/$mois/$annee - $heure:$minute:$seconde";
}

#================================================
# But : Centrer un widget automatiquement
#================================================
sub centrer_widget {
  my ($widget) = @_;

  # Height and width of the screen
  my $largeur_ecran = $widget->screenwidth();
  my $hauteur_ecran = $widget->screenheight();

  # update le widget pour récupérer les vraies dimensions
  $widget->update;
  my $largeur_widget = $widget->width;
  my $hauteur_widget = $widget->height;

  # On centre le widget en fonction de la taille de l'écran
  my $nouvelle_largeur = int( ( $largeur_ecran - $largeur_widget ) / 2 );
  my $nouvelle_hauteur = int( ( $hauteur_ecran - $hauteur_widget ) / 2 );
  $widget->geometry( $largeur_widget . "x" . $hauteur_widget . "+$nouvelle_largeur+$nouvelle_hauteur" );

  $widget->update;

  return;
}
```

Maintenant, regardons la procédure « recherchez_fichier_tk » qui est exécutée lorsque l'utilisateur lance la recherche de fichiers.

Procédure recherchez_fichier_tk

```
sub recherchez_fichier_tk {

    if ( not defined $motif_recherche or not defined $repertoire_emplacement or ! -d $repertoire_emplacement ) {
      return;
    }
    $etat_recherche = "Liste des fichiers en cours";

    # On lui indique la procédure à appeler
    $nom_fonction = "recherchez_fichier";

    # On lui donne les arguments
    @arguments_thread = ($motif_recherche, $repertoire_emplacement);

    # On va demander au thread de bosser
    $thread_travail   = 1;

  return;
}
```

Dans la procédure ci-dessus, on indique notre variable partagée : le nom de la procédure à appeler.

```
# On lui indique la procédure à appeler
    $nom_fonction = "recherchez_fichier";
```

On fait de même avec les bons arguments à passer à la procédure « recherchez_fichier » :

```
# On lui donne les arguments
    @arguments_thread = ($motif_recherche, $repertoire_emplacement);
```

Ensuite, nous demandons à notre processus léger de commencer à travailler.

```
# On va demander au thread de bosser
    $thread_travail   = 1;
```

Le fait de pointer la variable **$thread_travail** à 1 permettra au thread (dans le while) de lancer la recherche. Petit rappel :

notre_processus_leger

```
#================================================
# notre_processus_leger
#================================================
sub notre_processus_leger {

  # Tourne en rond
  while (1) {

    # demande au thread de travailler
    if ( $thread_travail == 1 ) {

      # Lance la procédure
      $fonctions_a_lancer_dans_thread{$nom_fonction}->(@arguments_thread);

      # demande au thread de dormir
      $thread_travail = 0;
    }

    # Terminer le thread
    last if ( $tuer_thread == 1 );
    sleep 0.5;
  }
  return;
}
```

La fonction de recherche lancée est la suivante :

Procédure recherchez_fichier

```
sub recherchez_fichier {
  my ($motif_recherche, $repertoire_emplacement)
= @_;

  # Recherchons le fichier
  my $fichier_trouve = 0;
  find(
    { wanted => sub {
        if ( $_ =~ m{$motif_recherche}i ) {
          $fichier_trouve++;
          print "$fichier_trouve-
$File::Find::name\n";
        }
      }
    },
    $repertoire_emplacement
  );

  return $fichier_trouve;
}
```

Voilà ! A ce stade, notre programme utilisant un thread fonctionne. Pour avoir une vue globale, le voici dans son intégralité :

Programme recherche_tk_thread.pl

```
#!/usr/bin/perl
#================================================
# Auteur : djibril
# Date   : 03/07/2011 20:35:16
# But    : Script Perl/Tk utilisant des threads
pour rechercher nos fichiers
#================================================
use warnings;
use strict;

use Tk;                          # Pour créer
notre GUI
use Tk::LabFrame;
use File::Find;
use threads;                     # Pour créer nos
threads
use threads::shared;             # Pour partager
nos données entre threads
use Time::HiRes qw( sleep );     # Pour faire des
sleeps < à une seconde

# Contient les fonctions à appeler dans le thread
si besoin
my %fonctions_a_lancer_dans_thread = (
  'recherchez_fichier' => \&recherchez_fichier,
);

#==================================
# Threads et variables partagées
#==================================
my $tuer_thread : shared;          # Permet de
tuer le thread proprement
my $nom_fonction : shared;         # Contient le
nom de la fonction à appeler
my $thread_travail : shared;       # Contient la
valeur permettant au thread de lancer une
procédure
my @arguments_thread : shared;     # Contient les
arguments à passer à une éventuelle procédure
my @resultat_fonction : shared;    # Contient le
résultat des fonctions lancées dans le thread
$thread_travail = 0;               # 0 : thread
ne fait rien, 1 : il bosse
$tuer_thread    = 0;               # 0 : thread
en vie, 1 : thread se termine

# Création du thread
my $thread = threads->create(
\&notre_processus_leger );

#==================================
# Début du code principal Perl Tk
#==================================
my $fenetre = new MainWindow(
  -title      => 'Recherche de fichiers',
  -background => 'white',
);
$fenetre->protocol( "WM_DELETE_WINDOW",
\&fermer_application );

# Affichage de l'heure
my $date_heure = date();
my $label_date = $fenetre->Label(
  -textvariable => \$date_heure,
  -background   => 'white',
  -font         => '{Arial} 16 {bold}',
)->pack(qw/ -pady 20 /);

# État de la recherche du fichier
my $etat_recherche = 'Aucune recherche en cours';
my $label_etat     = $fenetre->Label(
  -textvariable => \$etat_recherche,
  -background   => 'white',
  -foreground   => 'blue',
  -font         => '{Arial} 12 {bold}',
)->pack(qw/ -pady 20 /);

# Cadre de recherche
my $cadre = $fenetre->LabFrame(
  -label      => 'Cadre de recherche',
  -background => 'white',
)->pack(qw/ -pady 20 -padx 20 /);

my ( $motif_recherche, $repertoire_emplacement );
my $label1 = $cadre->Label( -text => 'Nom du
fichier à trouver : ', -background => 'white' );
my $entry_nom_fichier = $cadre->Entry(
-textvariable => \$motif_recherche );
my $label2 = $cadre->Label( -text => 'Emplacement
: ', -background => 'white' );
my $entry_emplacement = $cadre->Entry(
-textvariable => \$repertoire_emplacement );

my $bouton_emplacement = $cadre->Button(
  -text    => '...',
  -command => sub {
    $repertoire_emplacement = $cadre-
>chooseDirectory(
      -title     => 'Sélectionner un
emplacement',
      -mustexist => 1,
    );
  },
);

# Affichage d'un bouton pour rechercher un
fichier
my $bouton = $cadre->Button(
  -text    => 'Recherchez un fichier',
  -command => [ \&recherchez_fichier_tk ],
```

```
  -font    => '{Arial} 14 {bold}',
);

$label1->grid( $entry_nom_fichier, '-',
-sticky => 'nw' );
$label2->grid( $entry_emplacement,
$bouton_emplacement, -sticky => 'nw' );
$bouton->grid( '-',                '-',
qw/ -padx 10 -pady 10 / );

# Centrer ma fenêtre
centrer_widget($fenetre);

# Toutes les secondes, la date et l'heure
évoluent
$fenetre->repeat( 1000, sub { $date_heure =
date(); } );

MainLoop;

#================================================
# notre_processus_leger
#================================================
sub notre_processus_leger {

  # Tourne en rond
  while (1) {

    # demande au thread de travailler
    if ( $thread_travail == 1 ) {

      # Lance la procédure
      $fonctions_a_lancer_dans_thread{$nom_fonctio
n}->(@arguments_thread);

      # demande au thread de dormir
      $thread_travail = 0;
    }

    # Terminer le thread
    last if ( $tuer_thread == 1 );
    sleep 0.5;
  }
  return;
}

sub fermer_application {

  # Demande au thread de se terminer
  $tuer_thread = 1;

  # On attend que le thread se termine proprement
  $thread->detach();

  exit;
}

sub recherchez_fichier_tk {
    if ( not defined $motif_recherche or not
defined $repertoire_emplacement or ! -d
$repertoire_emplacement ) {
      return;
    }
    $etat_recherche = "Liste des fichiers en
cours";

    # On lui indique la procédure à appeler
    $nom_fonction = "recherchez_fichier";

    # On lui donne les arguments
    @arguments_thread = ($motif_recherche,
$repertoire_emplacement);

    # On va demander au thread de bosser
    $thread_travail   = 1;

  return;
}

sub recherchez_fichier {
  my ($motif_recherche, $repertoire_emplacement)
= @_;

  # Recherchons le fichier
  my $fichier_trouve = 0;
  find(
    { wanted => sub {
        if ( $_ =~ m{$motif_recherche}i ) {
          $fichier_trouve++;
          print "$fichier_trouve-
$File::Find::name\n";
        }
      }
    },
    $repertoire_emplacement
  );

  return $fichier_trouve;
}

#================================================
# But : Obtenir la date et l'heure
#================================================
sub date {
  my $time = shift || time;    #$time par defaut
vaut le time actuel
  my ( $seconde, $minute, $heure, $jour, $mois,
$annee, $jour_semaine, $jour_annee,
$heure_hiver_ou_ete )
    = localtime($time);
  $mois  += 1;
  $annee += 1900;

  # On rajoute 0 si le chiffre est compris entre
1 et 9
  foreach ( $seconde, $minute, $heure, $jour,
$mois, $annee ) { s/^(\d)$/0$1/; }
  return "$jour/$mois/$annee - $heure:$minute:
$seconde";
}

#================================================
# But : Centrer un widget automatiquement
#================================================
sub centrer_widget {
  my ($widget) = @_;

  # Height and width of the screen
  my $largeur_ecran = $widget->screenwidth();
  my $hauteur_ecran = $widget->screenheight();

  # update le widget pour récupérer les vraies
dimensions
  $widget->update;
  my $largeur_widget = $widget->width;
  my $hauteur_widget = $widget->height;

  # On centre le widget en fonction de la taille
de l'écran
  my $nouvelle_largeur = int( ( $largeur_ecran -
```

```
$largeur_widget ) / 2 );
  my $nouvelle_hauteur = int( ( $hauteur_ecran -
$hauteur_widget ) / 2 );
  $widget->geometry( $largeur_widget . "x" .
$hauteur_widget . "+$nouvelle_largeur+
$nouvelle_hauteur" );

  $widget->update;

  return;
}
```

N'hésitez à relire et lancer le programme pour comprendre son fonctionnement. Mais nous n'avons pas fini !
A ce stade, voici quelques remarques.

Avantages :

- lorsque l'on clique sur le bouton « Recherchez un fichier », la fenêtre n'est plus figée ;
- l'heure s'affiche normalement et régulièrement ;
- le thread ne bogue plus et ne s'arrête pas de façon brusque ;
- à la fermeture du script, on fait appel à la méthode **detach** sans souci. On aurait pu également appeler la méthode **join**.

Inconvénients :

- pour le moment, on ne sait pas comment récupérer le résultat de la procédure lancée dans le thread ;
- dans le thread principal, on ne sait pas concrètement quand le processus léger est terminé ;
- l'utilisateur peut cliquer sur le bouton « Recherchez un fichier » alors que la recherche est encore en cours.

On a quand même déjà beaucoup d'avantages par rapport aux inconvénients, non ? Nous allons maintenant voir comment on peut améliorer notre programme.

Il est important que vous soyez bien familier avec les notions de références en Perl pour cette partie de l'article.

Pour récupérer les résultats des procédures lancées dans notre thread, on a prévu une variable partagée.

```
my @resultat_fonction : shared;    # Contient le
résultat des fonctions lancées dans le thread
```

Nous allons l'utiliser dans notre procédure « Recherchez un fichier » en modifiant la ligne de code suivante :

Procédure notre_processus_leger
```
# Lance la procédure
      $fonctions_a_lancer_dans_thread{$nom_fonctio
n}->(@arguments_thread);
```

comme suit :

Procédure notre_processus_leger
```
# Lance la procédure
      my @resultat =
$fonctions_a_lancer_dans_thread{$nom_fonction}-
>(@arguments_thread);
```

On peut ainsi récupérer dans un premier temps le résultat retourné par la procédure lancée. Vous allez sûrement vous demander pourquoi on n'a pas tout simplement écrit :

perl
```
# Lance la procédure
      my @resultat_fonction =
$fonctions_a_lancer_dans_thread{$nom_fonction}-
>(@arguments_thread);
```

Si vous mettez directement le résultat dans **@resultat_fonction**, le code sera bon tant que votre procédure ne retourne que des scalaires, un tableau de scalaires partagés ou des références de tableaux (ou hash) partagées. C'est bien expliqué dans la documentation du module **threads::shared**.
Si ce n'est pas le cas, vous obtiendrez un message d'erreur du type *Thread 1 terminated abnormally: Invalid value for shared scalar at ...* et le thread s'arrêtera. Donc, prenons tout de suite de bonnes habitudes en utilisant la méthode **shared_clone** (du module thread) qui prend en argument une référence de tableau ou hash et copie tous les éléments non partagés.

NB : La méthode **shared_clone** retourne une référence de hash ou de tableau.

```
my $RefHash = shared_clone( \@ARRAY); # =>
retourne une référence de tableau
my $RefARRAY = shared_clone( \%HASH); # =>
retourne une référence de hash
```

Dans tous les cas, on utilise la méthode **shared_clone**. Comme elle nous retourne une référence, on va partager une variable s'appelant **$ref_resultat_fonction** à la place de **@resultat_fonction**, ce qui nous donne :

Procédure notre_processus_leger
```
#================================================
...
my $ref_resultat_fonction : shared; # Contient le
résultat des fonctions lancées dans le thread
...
...

#================================================
# notre_processus_leger
#================================================
sub notre_processus_leger {

  # Tourne en rond
  while (1) {

    # demande au thread de travailler
    if ( $thread_travail == 1 ) {

      # Lance la procédure
      my @resutat =
$fonctions_a_lancer_dans_thread{$nom_fonction}-
>(@arguments_thread);
      $ref_resultat_fonction = shared_clone(
\@resutat);

      # demande au thread de dormir
      $thread_travail = 0;
    }

    # Terminer le thread
```

```
    last if ( $tuer_thread == 1 );
    sleep 0.5;
  }
  return;
}
```

Notre variable contient maintenant à chaque fois le résultat de notre procédure.
Nous déterminons le moment où la procédure lancée par le processus léger est terminée et affichons le résultat. On empêche également un autre clic sur le bouton « Recherchez un fichier » tant que la recherche est en cours. Modifions notre procédure « recherchez_fichier_tk » :

Procédure recherchez_fichier_tk

```
sub recherchez_fichier_tk {
    if ( not defined $motif_recherche or not
defined $repertoire_emplacement or ! -d
$repertoire_emplacement ) {
      return;
    }

    # On désactive le bouton ListerFichiers
    $bouton->configure(-state => 'disabled');

    $etat_recherche = "Liste des fichiers en
cours";

    # On lui indique la procédure à appeler
    $nom_fonction = "recherchez_fichier";

    # On lui donne les arguments
    @arguments_thread = ($motif_recherche,
$repertoire_emplacement);

    # On va demander au thread de bosser
    $thread_travail   = 1;

  my $id;
  $id = $fenetre->repeat( 500, sub {
    if ( $thread_travail == 0 ) {
      # Thread terminé t on affiche le resulat
      my $nombre_fichier =
$ref_resultat_fonction->[0];
      $etat_recherche = "$nombre_fichier
fichier(s) trouvé(s)";
      $bouton->configure(-state => 'normal');
      $id->cancel;
    }
  } );

  return;
}
```

Explication :

Une fois que le thread se met à travailler, on désactive le bouton. Ensuite, via la méthode Tk **repeat**, on lance un code Perl qui vérifie (sans toutefois bloquer l'interface) toutes les 500 millisecondes si le processus léger a terminé sa tâche ou non en vérifiant que la variable **$thread_travail** est égale à 0 ou pas. Si la variable est à zéro, la tâche est terminée, on modifie le message qui sera affiché et on réactive le bouton et détruit l'événement Tk.

Voilà, cette fois notre programme est terminé et le voici :

Programme final : recherche_tk_thread_final

```
#!/usr/bin/perl
#=============================================
# Auteur : djibril
# Date   : 03/07/2011 14:08:50
# But    : Script Perl/Tk utilisant des threads
pour rechercher nos fichiers
#=============================================
use warnings;
use strict;

use Tk;    # Pour créer notre GUI
use Tk::LabFrame;
use File::Find;
use threads;                    # Pour créer nos
threads
use threads::shared;            # Pour partager
nos données entre threads
use Time::HiRes qw( sleep );    # Pour faire des
sleeps < à une seconde

# Contient les fonctions à appeler dans le thread
si besoin
my %fonctions_a_lancer_dans_thread =
( 'recherchez_fichier' =>
\&recherchez_fichier, );

#==================================
# Threads et variables partagées
#==================================
my $tuer_thread : shared;              # Permet
de tuer le thread proprement
my $nom_fonction : shared;             # Contient
le nom de la fonction à appeler
my $thread_travail : shared;           # Contient
la valeur permettant au thread de lancer une
procédure
my @arguments_thread : shared;         # Contient
les arguments à passer à une éventuelle procédure
my @resultat_fonction : shared;        # Contient
le résultat des fonctions lancées dans le thread
my $ref_resultat_fonction : shared;    # Contient
le résultat des fonctions lancées dans le thread

$thread_travail = 0;                   # 0 :
thread ne fait rien, 1 : il bosse
$tuer_thread    = 0;                   # 0 :
thread en vie, 1 : thread se termine

# Création du thread
my $thread = threads->create(
\&notre_processus_leger );

#==================================
# Debut du code principal Perl Tk
#==================================
my $fenetre = new MainWindow(
  -title      => 'Recherche de fichiers',
  -background => 'white',
);
$fenetre->protocol( "WM_DELETE_WINDOW",
\&fermer_application );

# Affichage de l'heure
my $date_heure = date();
my $label_date = $fenetre->Label(
  -textvariable => \$date_heure,
  -background   => 'white',
  -font         => '{Arial} 16 {bold}',
)->pack(qw/ -pady 20 /);
```

```
# État de la recherche du fichier
my $etat_recherche = 'Aucune recherche en cours';
my $label_etat     = $fenetre->Label(
  -textvariable => \$etat_recherche,
  -background   => 'white',
  -foreground   => 'blue',
  -font         => '{Arial} 12 {bold}',
)->pack(qw/ -pady 20 /);

# Cadre de recherche
my $cadre = $fenetre->LabFrame(
  -label      => 'Cadre de recherche',
  -background => 'white',
)->pack(qw/ -pady 20 -padx 20 /);

my ( $motif_recherche, $repertoire_emplacement );
my $label1 = $cadre->Label( -text => 'Nom du
fichier à trouver : ', -background => 'white' );
my $entry_nom_fichier = $cadre->Entry(
-textvariable => \$motif_recherche );
my $label2 = $cadre->Label( -text => 'Emplacement
: ', -background => 'white' );
my $entry_emplacement = $cadre->Entry(
-textvariable => \$repertoire_emplacement );

my $bouton_emplacement = $cadre->Button(
  -text    => '...',
  -command => sub {
    $repertoire_emplacement = $cadre-
>chooseDirectory(
      -title     => 'Sélectionner un
emplacement',
      -mustexist => 1,
    );
  },
);

# Affichage d'un bouton pour rechercher un
fichier
my $bouton = $cadre->Button(
  -text    => 'Recherchez un fichier',
  -command => [ \&recherchez_fichier_tk ],
  -font    => '{Arial} 14 {bold}',
);

$label1->grid( $entry_nom_fichier, '-',
-sticky => 'nw' );
$label2->grid( $entry_emplacement,
$bouton_emplacement, -sticky => 'nw' );
$bouton->grid( '-',                '-',
qw/ -padx 10 -pady 10 / );

# Centrer ma fenêtre
centrer_widget($fenetre);

# Toutes les secondes, la date et l'heure
évoluent
$fenetre->repeat( 1000, sub { $date_heure =
date(); } );

MainLoop;

#================================================
# notre_processus_leger
#================================================
sub notre_processus_leger {

  # Tourne en rond
  while (1) {

    # demande au thread de travailler
    if ( $thread_travail == 1 ) {

      # Lance la procédure
      my @resutat =
$fonctions_a_lancer_dans_thread{$nom_fonction}-
>(@arguments_thread);
      $ref_resultat_fonction = shared_clone(
\@resutat );

      # demande au thread de dormir
      $thread_travail = 0;
    }

    # Terminer le thread
    last if ( $tuer_thread == 1 );
    sleep 0.5;
  }
  return;
}

sub fermer_application {

  # Demande au thread de se terminer
  $tuer_thread = 1;

  # On attend que le thread se termine proprement
  $thread->detach();

  exit;
}

sub recherchez_fichier_tk {
  if ( not defined $motif_recherche or not
defined $repertoire_emplacement or !-d
$repertoire_emplacement ) {
    return;
  }

  # On désactive le bouton ListerFichiers
  $bouton->configure( -state => 'disabled' );

  $etat_recherche = "Liste des fichiers en
cours";

  # On lui indique la procédure à appeler
  $nom_fonction = "recherchez_fichier";

  # On lui donne les arguments
  @arguments_thread = ( $motif_recherche,
$repertoire_emplacement );

  # On va demander au thread de bosser
  $thread_travail = 1;

  my $id;
  $id = $fenetre->repeat(
    500,
    sub {
      if ( $thread_travail == 0 ) {

        # Thread terminé, on affiche le résultat
        my $nombre_fichier =
$ref_resultat_fonction->[0];
        $etat_recherche = "$nombre_fichier
fichier(s) trouvé(s)";
        $bouton->configure( -state => 'normal' );
        $id->cancel;
      }
    }
```

```
  );

  return;
}

sub recherchez_fichier {
  my ( $motif_recherche,
$repertoire_emplacement ) = @_;

  # Recherchons le fichier
  my $fichier_trouve = 0;
  find(
    { wanted => sub {
        if ( $_ =~ m{$motif_recherche}i ) {
          $fichier_trouve++;
          print "$fichier_trouve-
$File::Find::name\n";
        }
        }
    },
    $repertoire_emplacement
  );

  return $fichier_trouve;
}

#================================================
# But : Obtenir la date et l'heure
#================================================
sub date {
  my $time = shift || time;    #$time par defaut
vaut le time actuel
  my ( $seconde, $minute, $heure, $jour, $mois,
$annee, $jour_semaine, $jour_annee,
$heure_hiver_ou_ete )
    = localtime($time);
  $mois  += 1;
  $annee += 1900;

  # On rajoute 0 si le chiffre est compris entre
1 et 9
  foreach ( $seconde, $minute, $heure, $jour,
$mois, $annee ) { s/^(\d)$/0$1/; }
  return "$jour/$mois/$annee - $heure:$minute:
$seconde";
}

#================================================
# But : Centrer un widget automatiquement
#================================================
sub centrer_widget {
  my ($widget) = @_;

  # Height and width of the screen
  my $largeur_ecran = $widget->screenwidth();
  my $hauteur_ecran = $widget->screenheight();

  # update le widget pour récupérer les vraies
dimensions
  $widget->update;
  my $largeur_widget = $widget->width;
  my $hauteur_widget = $widget->height;

  # On centre le widget en fonction de la taille
de l'écran
  my $nouvelle_largeur = int( ( $largeur_ecran -
$largeur_widget ) / 2 );
  my $nouvelle_hauteur = int( ( $hauteur_ecran -
$hauteur_widget ) / 2 );
  $widget->geometry( $largeur_widget . "x" .
$hauteur_widget . "+$nouvelle_largeur+
$nouvelle_hauteur" );

  $widget->update;

  return;
}
```

En résumé :

- la fenêtre n'est plus figée et l'heure s'affiche de manière régulière ;
- l'utilisateur ne peut pas cliquer sur le bouton lorsque la recherche est en cours ;
- on récupère proprement les résultats de notre listing de fichiers ;
- le thread ne s'arrête plus brusquement ;
- à la fermeture de la fenêtre Perl Tk, le thread est proprement détruit.

Voilà, vous savez maintenant comment utiliser les threads avec Perl/Tk ! Vous pouvez vous inspirer de ces programmes pour l'utilisation des threads. Adaptez-les à vos besoins ! Si vous voulez utiliser un seul thread pour pouvoir lancer diverses procédures, il vous suffit de modifier le hash **%fonctions_a_lancer_dans_thread**.

Exemples

```
my %fonctions_a_lancer_dans_thread = (
  'recherchez_fichier' => \&recherchez_fichier,
  'ZiperUnRepertoire'  => \&ZiperUnRepertoire,
  'CalculTresLong'     => \&CalculTresLong,
  'CalculTresTresLong' => \&CalculTresTresLong,
  ...
);
```

Dans notre exemple, on a attendu que le thread se rendorme pour poursuivre le script principal, mais ce n'est pas une obligation. Tout dépend de ce que vous voulez faire. Il faut juste faire attention *"à ne pas se mélanger les pinceaux"* et ne pas écraser les données par erreur. Il est possible de faire des choses plus complexes, tout est fonction de votre cahier des charges et de votre imagination. On a également décidé d'utiliser un thread, mais vous auriez pu en utiliser plusieurs, c'est toujours le même principe. A vous de bien définir ce que vous souhaitez, à penser à protéger les variables partagées si nécessaire via la méthode **lock** (du module **threads::shared**).

Pour conclure, voici quelques inconvénients à l'utilisation des threads avec Perl/Tk (et oui, il y en a quand même) :

- vous avez pu remarquer que l'on est obligé de les créer en début de script. Si l'on choisit d'en créer un seul, il nous sera impossible d'en rajouter ;
- si pour différentes raisons, la procédure lancée par votre thread produit un **die**, votre thread sera détruit. Si vous n'en aviez qu'un, votre script ne pourra donc plus fonctionner correctement ;
- il peut être important de tester que nos threads sont toujours en vie ;
- pour finir, vous avez pu constater que l'utilisation des threads en Perl/Tk n'est pas très évidente, il faut se creuser les méninges.

Quoi qu'il en soit, un bon algorithme est nécessaire pour utiliser au mieux les threads et construire une application Perl Tk puissante.

4. Téléchargement des scripts

Tous les programmes conçus pour cet article sont téléchargeables en une fois ici : Lien 182. N'hésitez pas à les tester pour mieux comprendre le fonctionnement des threads ou du module Win32::Process avec Perl/Tk.

5. Liens utiles

Quelques références sur Perl/Tk et les threads :

1. threads (CPAN) : Lien 176 ;
2. threads::shared (CPAN) : Lien 179 ;
3. Win32::Process (CPAN) : Lien 177 ;
4. Tutoriel sur les threads en Perl : Lien 180 ;
5. http://www.perltk.org/ : Lien 183 ;
6. **perldoc perlthrtut**.

6. Conclusion

Dans cet article, nous avons appris comment ne pas figer une application de différentes façons : utiliser la méthode Tk (**update**), utiliser le module **Win32::Process** (uniquement sous Windows) et les threads. Pensez toujours à utiliser la méthode la plus simple tant que vous pouvez (notamment avec la méthode update), ou Win32::Process et en dernier recours, si vous n'avez vraiment pas le choix car vous utilisez un module externe pour effectuer de longs calculs, alors pensez aux threads. Gardez à l'esprit qu'en cas de "die" dans un thread, ce dernier meurt. De plus, il faut bien réfléchir au nombre de threads à créer en début de script, à la façon dont on souhaite protéger les données partagées...

N.B. Il existe sûrement d'autres modules externes permettant de lancer des programmes externes ou code Perl sans figer l'application. Je pense notamment aux modules POE::* (Lien 184). Si certains lecteurs souhaitent m'aider à compléter cet article en utilisant ce module ou un autre, j'en serais ravi. Maintenant que vous êtes bien armé, à vous de jouer !

Retrouvez l'article de djibril en ligne : Lien 185

Liens

Lien 01 : http://javasearch.developpez.com/j2se/1.6.0/docs/api/java/util/ServiceLoader.html
Lien 02 : http://ydisanto.developpez.com/tutoriels/java/services/
Lien 03 : http://jdk7.java.net/download.html
Lien 04 : http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html
Lien 05 : http://blogs.oracle.com/darcy/
Lien 06 : http://download.java.net/jdk7/docs/api/java/nio/file/package-summary.html
Lien 07 : http://download.java.net/jdk7/docs/api/java/nio/file/WatchService.html
Lien 08 : http://today.java.net/pub/a/today/2008/07/03/jsr-203-new-file-apis.html
Lien 09 : http://download.java.net/jdk7/docs/api/java/lang/invoke/package-summary.html
Lien 10 : http://download.java.net/jdk7/docs/api/java/util/concurrent/ForkJoinPool.html
Lien 11 : http://download.oracle.com/javase/7/docs/api/index.html?java/util/Objects.html
Lien 12 : http://blog.developpez.com/adiguba/p8597/java/7-dolphin/java-util-objects-simple-et-efficace/
Lien 13 : http://www.developpez.net/forums/d1105728/java/general-java/7-7-cest-java-7-a/
Lien 14 : http://mathias-seguy.developpez.com/cours/android/construction-ihm-dynamique/
Lien 15 : http://dsilvera.developpez.com/tutoriels/android/creer-page-login-verifier-identification-base-donnees-script-php/#LII-E
Lien 16 : http://dsilvera.developpez.com/tutoriels/android/creer-page-login-verifier-identification-base-donnees-script-php/
Lien 17 : http://help.eclipse.org/indigo/index.jsp?topic=%2Forg.eclipse.platform.doc.user%2FwhatsNew%2Fplatform_whatsnew.html
Lien 18 : http://wiki.eclipse.org/Eclipse_DemoCamps_Indigo_2011/Grenoble
Lien 19 : http://wiki.eclipse.org/Eclipse_DemoCamps_Indigo_2011/Nantes
Lien 20 : http://toulibre.org/eclipseparty
Lien 21 : http://www.eclipse.org/downloads/
Lien 22 : http://www.developpez.net/forums/d1099346/environnements-developpement/eclipse/eclipse-3-7-indigo-disponible/
Lien 23 : http://g-rossolini.developpez.com/tutoriels/php/cours/?page=concepts#LVI-G
Lien 24 : http://php.net/fr/addslashes
Lien 25 : http://php.net/fr/mysql_real_escape_string
Lien 26 : http://php.net/fr/pg_escape_string
Lien 27 : http://php.net/fr/escapeshellcmd
Lien 28 : http://php.net/fr/escapeshellarg
Lien 29 : http://php.net/fr/quotemeta
Lien 30 : http://php.net/fr/htmlentities
Lien 31 : http://php.net/fr/htmlspecialchars
Lien 32 : http://php.net/fr/urlencode
Lien 33 : http://php.net/fr/rawurlencode
Lien 34 : http://php.net/fr/base64_encode
Lien 35 : http://php.net/fr/magic_quotes_gpc
Lien 36 : http://php.net/fr/magic_quotes_sybase
Lien 37 : http://php.net/fr/magic_quotes_runtime
Lien 38 : http://dico.developpez.com/html/956-Langages-deprecated.php
Lien 39 : http://php.net/fr/html_entity_decode
Lien 40 : http://php.net/fr/htmlspecialchars_decode
Lien 41 : http://dico.developpez.com/html/883-Securite-cross-site-scripting.php
Lien 42 : http://www.php.net/manual/fr/faq.html.php#faq.html.encoding
Lien 43 : http://www.faqs.org/rfcs/rfc3986.html
Lien 44 : http://www.php.net/manual/fr/faq.html.php#faq.html.javascript-variable
Lien 45 : http://php.net/fr/mysqli_real_escape_string
Lien 46 : http://dico.developpez.com/html/884-Securite-injection-SQL.php
Lien 47 : http://php.net/fr/function.exec
Lien 48 : http://php.net/fr/system
Lien 49 : http://php.net/fr/passthru
Lien 50 : http://php.net/fr/proc_open
Lien 51 : http://php.net/fr/shell_exec
Lien 52 : http://php.net/fr/language.operators.execution
Lien 53 : http://php.net/fr/preg_quote
Lien 54 : http://php.net/fr/base64_decode
Lien 55 : http://php.net/fr/pdostatement.execute
Lien 56 : http://www.slideshare.net/kkotowicz/sql-injection-complete-walktrough-not-only-for-php-developers
Lien 57 : http://php.net/manual/fr/book.filter.php
Lien 58 : http://php.net/fr/filter_input_array
Lien 59 : http://jcrozier.developpez.com/tutoriels/web/php/filtres-securite/
Lien 60 : http://fladnag.developpez.com/upload/In.class.php.txt
Lien 61 : http://fladnag.developpez.com/tutoriels/php/formatage-donnees-php/
Lien 62 : http://buildinternet.com/2008/12/introduction-to-charts-with-flex-3/
Lien 63 : https://freeriatools.adobe.com/
Lien 64 : http://kalyparker.developpez.com/articles/flex/Diagrammeflex/fichiers/xml-assets.zip
Lien 65 : http://buildinternet.com/2008/12/xml-basics-with-flex-3/
Lien 66 : http://livedocs.adobe.com/flex/3/html/help.html?content=charts_types_01.html
Lien 67 : http://buildinternet.com/2008/12/flex-3-basics-introduction-to-data-binding/
Lien 68 : http://www.buildinternet.com/live/flexcharts/source.zip
Lien 69 : http://www.buildinternet.com/live/flexcharts/charts.html
Lien 70 : http://kalyparker.developpez.com/articles/flex/diagrammeflex/
Lien 71 : http://dev.w3.org/html5/spec/Overview.html#the-xhtml-syntax
Lien 72 : http://tcuvelier.developpez.com/tutoriels/web-semantique/rdfa/introduction/
Lien 73 : http://dev.w3.org/html5/vocabulary/
Lien 74 : http://www.data-vocabulary.org/
Lien 75 : http://tcuvelier.developpez.com/tutoriels/web-semantique/html5-microdonnees/introduction/
Lien 76 : http://googleblog.blogspot.com/2011/06/introducing-schemaorg-search-engines.html
Lien 77 : http://www.ysearchblog.com/2011/06/02/introducing-schema-org-a-collaboration-on-structured-data/

Lien 78 : http://www.bing.com/community/site_blogs/b/search/archive/2011/06/02/bing-google-and-yahoo-unite-to-build-the-web-of-objects.aspx
Lien 79 : http://schema.org/
Lien 80 : http://web-semantique.developpez.com/actu/32826/Web-semantique-Microsoft-Google-et-Yahoo-collaborent-sur-Scheme-org-un-microformat-pour-structurer-le-HTML-aux-moteurs-de-recherche/
Lien 81 : http://jplu.developpez.com/tutoriels/web-semantique/introduction/
Lien 82 : http://dublincore.org/
Lien 83 : http://www.foaf-project.org/
Lien 84 : http://wiki.dbpedia.org/Ontology
Lien 85 : http://www.data-vocabulary.org/
Lien 86 : https://developers.facebook.com/docs/opengraph/
Lien 87 : http://web-semantique.developpez.com/tutoriels/jena/arq/introduction-sparql/
Lien 88 : http://web-semantique.developpez.com/tutoriels/jena/introduction-rdf/
Lien 89 : http://tcuvelier.developpez.com/tutoriels/web-semantique/html5-microdonnees/introduction/
Lien 90 : http://tcuvelier.developpez.com/tutoriels/web-semantique/rdfa/introduction/
Lien 91 : http://dev.w3.org/html5/md/
Lien 92 : http://dev.w3.org/html5/md/#rdf
Lien 93 : http://purl.org/dc/terms/title
Lien 94 : http://jplu.developpez.com/tutoriels/web-semantique/apprendre-rdf-par-exemple-avec-foaf/
Lien 95 : http://www.w3.org/1999/xhtml/vocab#
Lien 96 : http://purl.org/dc/terms/source
Lien 97 : http://schema.org/docs/schemas.html
Lien 98 : http://schema.org/docs/full.html
Lien 99 : http://schema.org/Movie
Lien 100 : http://schema.org/Person
Lien 101 : http://www.google.com/support/webmasters/bin/answer.py?hl=fr&answer=66353
Lien 102 : http://schema.org/Thing
Lien 103 : http://schema.org/ItemAvailability
Lien 104 : http://dev.w3.org/html5/md/Overview.html
Lien 105 : http://schema.org/docs/gs.html
Lien 106 : http://schema.org/docs/extension.html
Lien 107 : http://tcuvelier.developpez.com/tutoriels/web-semantique/html5-microdonnees/schema-org/
Lien 108 : http://www.developpez.net/forums/d1119805/c-cpp/cpp/communaute/cpp11-cest-vote-unanime-approuver-nouvelle-normalisation-langage/
Lien 109 : http://www.youtube.com/user/EADStv?blend=1&ob=5#p/a/E2CF0F65A3AF26B1/1/Z1k8KqHq7Bk
Lien 110 : http://glc-lib.net/download.php
Lien 111 : http://www.developpez.net/forums/d1104930/c-cpp/bibliotheques/qt/outils/bibliotheques/glc_lib/glc_lib-2-2-0-sortie/
Lien 112 : http://www.qxorm.com/
Lien 113 : http://www.developpez.net/forums/d1100883/c-cpp/bibliotheques/qt/outils/bibliotheques/qxorm/qxorm-1-1-7-vient-sortir/
Lien 114 : http://lists.pyside.org/pipermail/pyside/2011-July/002648.html
Lien 115 : http://www.pyside.org/2011/07/pyside-1-0-5-and-no-name-was-given-that-day-python-for-qt-released/
Lien 116 : http://www.pyside.org/2011/07/pyside-1-0-5---and-no-name-was-given-that-day-packages-available-for-fremantle/
Lien 117 : http://developer.qt.nokia.com/wiki/Category:LanguageBindings::PySide::Downloads
Lien 118 : http://www.developpez.net/forums/d1014139/autres-langages/python-zope/gui/pyside-pyqt/sortie-pyside-1-0-5-a/#post6161230
Lien 119 : http://qt.developpez.com/doc/4.7/liste-des-pages-traduites/
Lien 120 : http://qt.developpez.com/doc/4.7/qgraphicsscene/
Lien 121 : http://qt.developpez.com/doc/4.7/qgraphicsview/
Lien 122 : http://qt.developpez.com/doc/4.7/qlistview/
Lien 123 : http://qt.developpez.com/doc/4.7/qtableview/
Lien 124 : http://qt.developpez.com/doc/4.7/qtreeview/
Lien 125 : http://qt.developpez.com/doc/4.7/qtabletevent/
Lien 126 : http://qt.developpez.com/doc/4.7/qsymbianevent/
Lien 127 : http://qt.developpez.com/doc/4.7/qpainterpath/
Lien 128 : http://qt.developpez.com/doc/4.7/classes/
Lien 129 : http://qt.developpez.com/doc/4.7/vues-d-ensemble/
Lien 130 : http://www.developpez.net/forums/d941430-2/c-cpp/bibliotheques/qt/documentation-qt-4-7-francais-arbre-classes/#post6135275
Lien 131 : http://qt.developpez.com/actu/32495/Sortie-de-Qt-4-8-Technology-Preview-avec-Lighthouse-la-couche-d-abstraction-du-systeme-graphique/
Lien 132 : http://qt.developpez.com/actu/32339/QNetworkAccessManager-fonctionnera-de-maniere-threadee-des-Qt-4-8-ameliorant-les-performances-de-QtWebKit-sans-y-toucher/
Lien 133 : http://labs.qt.nokia.com/2011/07/19/qt-4-8-beta-released/
Lien 134 : http://www.developpez.net/forums/d992306/c-cpp/bibliotheques/qt/qt-4-8-beta-sortie/#post6136500
Lien 135 : http://qt.developpez.com/doc/4.7/qtquick/
Lien 136 : http://www.developpez.net/forums/d891380/c-cpp/cpp/cpp0x-draft-final-ete-vote/
Lien 137 : http://en.wikipedia.org/wiki/C++0x
Lien 138 : http://www2.research.att.com/~bs/C++0xFAQ.html
Lien 139 : http://labs.trolltech.com/page/Main_Page
Lien 140 : http://labs.qt.nokia.com/2011/05/26/cpp0x-in-qt/
Lien 141 : http://gcc.gnu.org/projects/cxx0x.html
Lien 142 : http://blogs.msdn.com/b/vcblog/archive/2010/04/06/c-0x-core-language-features-in-vc10-the-table.aspx
Lien 143 : http://qt.developpez.com/doc/4.7/qvector/
Lien 144 : http://qt.developpez.com/doc/4.7/qlist/
Lien 145 : http://qt.developpez.com/doc/4.7/qstringlist/
Lien 146 : http://qt.developpez.com/doc/4.7/implicit-sharing/
Lien 147 : http://www.cplusplus.com/reference/stl/vector/
Lien 148 : http://qt.developpez.com/doc/4.7/qimage/
Lien 149 : http://qt.developpez.com/doc/4.7/qtglobal/#foreach
Lien 150 : http://www.boost.org/doc/libs/1_46_0/doc/html/foreach.html
Lien 151 : http://qt.developpez.com/doc/4.7/qvector/#begin
Lien 152 : http://qt.developpez.com/doc/4.7/qvector/#end
Lien 153 : http://qt.developpez.com/doc/4.7/programmation-concurrente/
Lien 154 : http://qt.developpez.com/doc/4.7/QtConcurrentRun/
Lien 155 : http://qt.developpez.com/doc/4.7/qtconcurrentmap/#map
Lien 156 : http://qt.developpez.com/doc/4.7/qtconcurrentmap/#mapped

Lien 157 : http://blogs.msdn.com/b/vcblog/archive/2010/04/06/c-0x-core-language-features-in-vc10-the-table.aspx
Lien 158 : http://gcc.gnu.org/projects/cxx0x.html
Lien 159 : http://qt-labs.developpez.com/cpp/cpp11/
Lien 160 : http://qt.developpez.com/doc/latest/qwidget.html
Lien 161 : http://qt.developpez.com/doc/latest/qobject.html
Lien 162 : http://qt.developpez.com/doc/latest/signalsandslots.html
Lien 163 : http://qt.developpez.com/doc/latest/qpushbutton.html
Lien 164 : http://qt.developpez.com/doc/latest/qapplication.html
Lien 165 : http://qt.developpez.com/doc/latest/qlayout.html
Lien 166 : http://qt.developpez.com/doc/latest/layout.html
Lien 167 : http://qt.developpez.com/doc/latest/QLineEdit
Lien 168 : http://qt.developpez.com/doc/latest/QLabel
Lien 169 : http://qt.developpez.com/doc/latest/QHBoxLayout
Lien 170 : http://tcuvelier.developpez.com/tutoriels/pyqt/signaux-slots-layouts/
Lien 171 : http://www.developpez.net/forums/d922999/autres-langages/pascal/defi-pascal-2010-systeme-chat-resultat/#post5208682
Lien 172 : ftp://ftp-developpez.com/mick605/articles/creation-chat/systeme-chat-simple.zip
Lien 173 : ftp://ftp-developpez.com/mick605/articles/creation-chat/systeme-chat-v2.zip
Lien 174 : http://mick605.developpez.com/articles/creation-chat/
Lien 175 : http://perl.developpez.com/
Lien 176 : http://search.cpan.org/search?query=threads&mode=all
Lien 177 : http://search.cpan.org/search?query=Win32%3A%3AProcess&mode=all
Lien 178 : http://perl.developpez.com/faq/perl/?page=sectionC4#sectionC41
Lien 179 : http://search.cpan.org/search?query=threads%3A%3Ashared&mode=all
Lien 180 : http://perl.enstimac.fr/DocFr/perlthrtut.html
Lien 181 : http://cpansearch.perl.org/src/SREZIC/Tk-804.028/ToDo
Lien 182 : ftp://ftp-developpez.com/djibril/tutoriels/perl/application-perl-tk-non-figee-threads-win32/fichiers/perl-tk-threads-win32-poe.zip
Lien 183 : http://www.perltk.org/
Lien 184 : http://search.cpan.org/search?query=POE%3A%3A*&mode=all
Lien 185 : http://djibril.developpez.com/tutoriels/perl/application-perl-tk-non-figee-threads-win32/