Developpez

Le Mag

Édition de avril - mai 2012.
Numéro 39.
Magazine en ligne gratuit.
Diffusion de copies conformes à l'original autorisée.
Réalisation : Alexandre Pottiez
Rédaction : la rédaction de Developpez
Contact : magazine@redaction-developpez.com

Sommaire

Qt	Page 2
C++	Page 8
Java	Page 13
Eclipse	Page 20
Android	Page 27
DotNet	Page 29
Web sémantique	Page 37
Perl	Page 44
MS Office	Page 47
Business Intelligence	Page 53
Ruby on Rails	Page 58
Développement Web	Page 65
Liens	Page 69

Éditorial

Retrouvez les meilleurs articles, news, billets blog du site Developpez.com, compilés dans ce nouveau numéro du magazine du club des professionnels de l'informatique.

Profitez-en bien !

La rédaction

Article C++

Commencer facilement avec Boost Graph

Ce tutoriel est une introduction pratique à Boost Graph permettant de prendre en main facilement les fonctionnalités de base.

par **Guillaume Belz**
Page 8

Article Business Intelligence

Qu'est-ce que l'informatique décisionnelle ?

Ce tutoriel, destiné aux débutants, a pour objectif de vous expliquer ce qu'est l'informatique décisionnelle.

par **Michael Tranchant**
Page 53

Qt Creator 2.5 beta est sorti

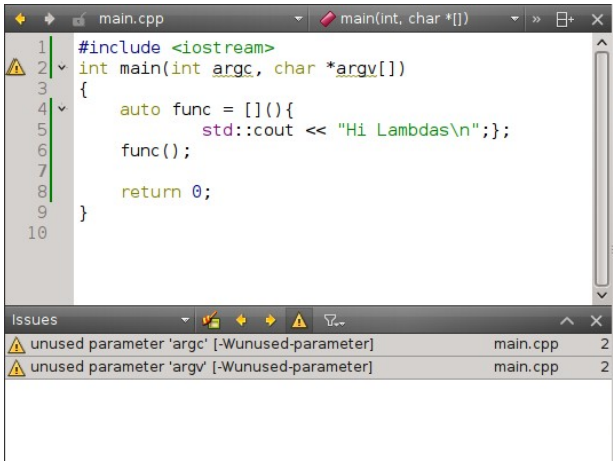
Suite à la sortie de Qt Creator 2.5 beta, il est grand temps de faire le tour de quelques nouveautés, sans toutes les passer en revue.

C++11

Publié en septembre dernier (Lien 01), le standard ISO C++11 se doit d'avoir un meilleur support dans l'EDI ; notamment, on trouvera les mots-clés nullptr, constexpr, static_assert, noexcept et auto, ainsi que les espaces de noms en ligne et les lambdas (partiellement).

De même, quelques nouvelles actions de refactorisation sont disponibles : insertion d'un #include pour les identifiants indéfinis, extraction de fonction, réarrangement de liste de paramètres, synchronisation des signatures (changer le nom d'un paramètre dans la déclaration répercutera la modification dans le corps).

Les débogueurs utilisés sous Linux et avec MinGW affichent également de manière améliorée certains des nouveaux types de base de C++11, tels que std::shared_ptr, std::unique_ptr, std::array. De plus, Qt Creator propose la même chose pour std::complex, boost::posix_time::ptime, boost::posix_time::time_duration, boost::gregorian::date. De manière plus fréquente, Qt Creator devinera correctement le type dynamique des pointeurs et en affichera mieux le contenu.

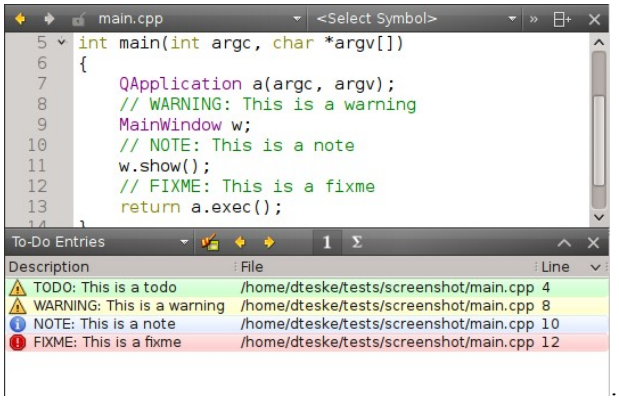


Statistiques

Un jour avant la sortie de Qt Creator 2.4 beta, le projet fou de l'open gouvernance pour Qt a été lancé (Lien 02). Notamment, cela a instauré l'utilisation de Gerrit pour la revue de code ; depuis lors, pas moins de septante-huit personnes ont soumis des patches à Qt Creator !

Plug-in TODO

Notamment, l'une des plus grosses contributions a été un plug-in TODO. Il lui manque encore un coup de polish, il est donc désactivé par défaut pour le moment, bien que déjà fort utilisable (activable dans Help > About plugins).



Plus petites améliorations

À côté de ces plus grandes nouveautés, d'autres petits changements sont susceptibles de simplifier fortement la vie aux utilisateurs de Qt Creator, comme un historique pour le copier-coller (Ctrl+Shift+V pour y naviguer).

Qt Creator 2.5 beta est téléchargeable depuis le wiki du Qt Project : (Lien 03)

(Depuis la publication de cette news, la version finale est sortie, consultez la news suivante pour plus d'information : Lien 04)

Commentez la news de Thibaut Cuvelier en ligne : Lien 05

PySide devient un add-on Qt

Le binding Python initié par Nokia est toujours disponible sous la même licence

Le Qt Project étant arrivé depuis quelques mois (Lien 06), rien de plus normal que de voir le binding Python initié par Nokia le rejoindre : ce projet est maintenant plus aligné avec le framework Qt et bénéficie de toute l'infrastructure mise en place (nouvel emplacement pour la

mailing list (Lien 07), le suivi des problèmes sur Jira (Lien 08), la méritocratie dans l'open gouvernance (Lien 09)).

Pour arriver à ce résultat, il a fallu, pour suivre ce modèle ouvert, désigner plusieurs personnes pour remplir les rôles de Maintainer et Approver pour chaque module. C'est ainsi que Marcelo Lira sera Maintainer pour API Extractor, Generatorrunner et Shiboken, Hugo Parente Lima pour PySide ; Paulo Alcantara sera Approver pour PySide. Tous les autres rôles sont officieux.

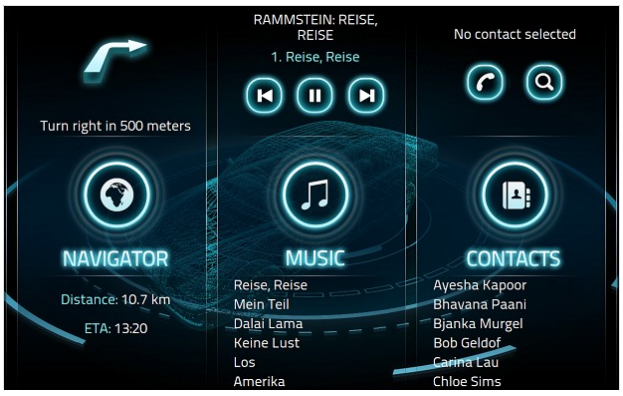
Du côté des utilisateurs, peu de changements ou pas : le projet est toujours disponible sous la même licence (c'est pour cela qu'il a été initié), l'API n'a évidemment pas été affectée, les mêmes outils sont toujours disponibles, certaines adresses ont changé. Dans les nouveautés, le dépôt Gitorious est devenu lecture seule, toute contribution devra passer par le système de *code review* Gerrit, comme tout Qt.

Commentez la news de Thibaut Cuvelier en ligne : Lien 10

Digia dévoile un nouveau concept d'interface homme-machine (IHM) en 3D pour les automobiles implémentée avec Qt

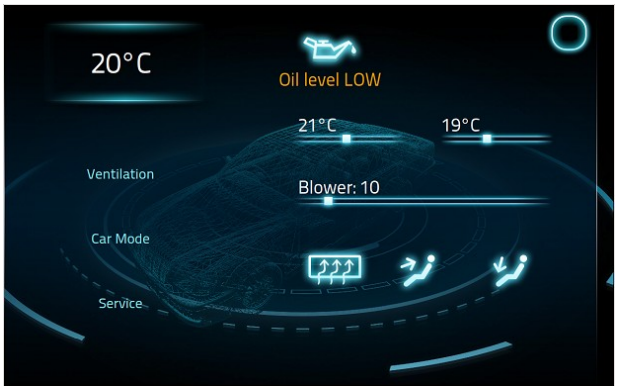
Aux Qt Developer Days 2011 de Munich, Digia Qt Commercial a montré un nouveau concept d'interface homme-machine (IHM) en 3D pour les automobiles pour la prochaine version commerciale de Qt 4.8. Le concept sera utilisé comme une nouvelle plateforme d'innovation pour faciliter l'intégration et la présentation de nouvelles idées technologiques.

Le principe d'interaction de l'IHM 3D est prévu pour la facilité et l'intuitivité de son utilisation. L'utilisateur est en mesure de contrôler l'interface utilisateur avec des gestes approximatifs au lieu de pointer précisément. Puisque le système met en avant automatiquement les informations les plus pertinentes suivant le contexte, il est facile de maîtriser le volume croissant d'informations fournies par le véhicule (rappels intelligents du système embarqué de navigation et des systèmes de contrôle de la circulation, etc.).



Avec ce nouveau concept d'IHM (des pages dédiées affichant des informations), l'utilisateur peut facilement naviguer par des gestes simples (haut/bas et gauche/droite). Chaque page identifie l'information la plus pertinente et la plus utilisée, il est donc facile d'obtenir un aperçu des sujets les plus communs du niveau principal. Pour des informations détaillées sur n'importe quel élément, l'utilisateur peut facilement naviguer dans la page et obtenir des informations complètes dans cette zone. En outre, le système est conçu pour être sensible au contexte et être capable d'orienter l'utilisateur vers l'information dont il a besoin.

Les technologies 3D, s'appuyant sur OpenGL avec l'API C++ QtOpenGL, facilitent la compréhension de la localisation visuelle pour l'entretien et les opérations sur le véhicule. Par exemple, les éléments habituels tels que la pression des pneus, les informations de diagnostic et les notifications de sécurité sont maintenant visualisées via le modèle 3D du véhicule : l'utilisateur comprendra d'un simple coup d'œil les indications du système. À l'avenir, d'autres systèmes (tels que le sonar, le régulateur de vitesse intelligent et la caméra de recul) seront intégrés.



Sur quelles plateformes peut-il tourner ?

Digia a présenté ce nouveau concept fonctionnant sous Linux sur du matériel embarqué basé sur un processeur ARM OMAP4 (Texas Instrument, cadencé à 1 GHz). Les parties 3D sont accélérées matériellement avec le GPU intégré du processeur. En exploitant les capacités multiplateformes de Qt Commercial, il est facile d'exécuter le même code sur un grand nombre d'autres plateformes matérielles et systèmes d'exploitation.

Le nouveau concept a été créé pour s'adapter à différents mécanismes d'entrée, y compris le contrôle direct par contact, le trackpad et la reconnaissance vocale. Digia vise à développer davantage le concept, qui sera plus tard mis à disposition sous forme de code source pour les licences Qt commerciales.

Commentez la news de Guillaume Belz en ligne : Lien 11

Sortie de Qt 5 alpha

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

La version 5 de Qt vient de sortir en version alpha. Cette version est la première version majeure depuis que Qt est devenu autonome avec la création du Qt Project. Beaucoup de personnes ont contribué à cette nouvelle version, pas uniquement des développeurs de chez Nokia. Les différents modules ont été regroupés en deux

catégories, les essentiels, installés par défaut, et les add-ons, installés à la demande. L'objectif de cette version alpha est de récupérer les retours des utilisateurs, principalement sur les modules essentiels.

Lars Knoll, le responsable en chef du projet Qt, a publié en mai dernier deux discussions sur les QtLabs pour présenter les approches choisies pour Qt 5 (voir les discussions **Thoughts about Qt 5 (Lien 12)** et **Responses to Qt 5 (Lien 13)**). La pensée directrice est résumée dans les phrases suivantes :

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

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

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

Le Qt Platform Abstraction Layer (QPA) : Lien 14

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

La réorganisation de la pile graphique

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

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

On note l'apparition de nouvelles classes :

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

L'architecture modulaire

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

Déplacer QWidget dans un module indépendant

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

Installation et compilation

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

- à partir des dépôts ppa (Linux) : Lien 15 ;
- à partir de Git : Building Qt 5 from Git (Lien 16) ;
- à partir des sources Qt 5.0 Alpha release en différents format (7z, tar.bz2, tar.gz, tar.xz, zip) (Lien 17) ;
- pour compiler : Qt 5 Alpha building instructions (Lien 18).

Passer de Qt 4 à Qt 5

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

Code :

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

Le script Perl `qtbase/bin/fixqt4headers.pl` met à jour les inclusions des fichiers d'en-tête.

Pour la création de plugins, les macros `Q_EXPORT_PLUGIN` et `Q_EXPORT_PLUGIN2` sont dépréciées et doivent être remplacées par la macro `Q_PLUGIN_METADATA`, qui permet de lire les informations sans devoir charger le plugin avec la fonction dlopen().

Que pensez-vous de la direction prise par Qt ?
Pensez-vous que Qt Quick et le QML prennent une place trop importante ou au contraire devraient se développer plus ?
Quelles sont les fonctionnalités que vous attendez le plus dans Qt 5 ?

Commentez la news de Guillaume Belz en ligne : Lien 19

Créer des applications avec un style Metro avec Qt

Comme déjà montré (Lien 20), les applications Qt Commercial en C++ et Qt Quick sont utilisables sans aucun problème ni modification sur la préversion pour développeurs de Windows 8. Ici, on montrera, avec plus de détails, comment des applications Qt peuvent adopter un style Metro. Les exemples seront des applications Qt normales dont le style aura été adapté pour ressembler à des applications Metro.

1. L'article original

Cet article est une adaptation en langue française de How to Create Qt Applications with Metro Style, de Sami Makkonen : Lien 21.

2. Introduction

Metro est le nom donné au nouveau langage de design d'interfaces utilisateur créé par Microsoft. Actuellement, Metro est utilisé comme base pour les Windows Phone 7 et, plus tard, pour l'interface de Windows 8. Pour récupérer l'essence du style Metro, on va créer un style Qt en utilisant plusieurs principes généraux de style et de design.

Comme expliqué dans le guide des principes généraux de design de Microsoft (Lien 22), le style Metro se focalise sur le contenu. Metro est prévu pour être simple et léger, libre d'éléments au look chromé, où le contenu est l'interface. Les principes-clés de design sont la cohérence, la typographie et le mouvement. Pour apporter une structure cohérente à l'interface, tous les items sont séparés en utilisant des marges cohérentes et les items sont généralement alignés à gauche. Metro utilise extensivement le texte et la typographie. Le texte brut est utilisé non seulement pour montrer le contenu, mais aussi comme marqueur indiquant la voie vers plus de contenu. Le type de police est Segoe (en fonction des plateformes) (Lien 23) - la famille et la taille de police sont cohérentes en fonction de l'usage (certaines tailles seront réservées aux titres, par exemple). Metro ne supporte pas la rognure de texte ou les affaiblissements. En lieu et place, tout le texte qui ne tient pas sur l'écran est mis sur la droite, par-delà l'écran. Comme mentionné, l'un des éléments-clés de Metro est le mouvement : des animations pour les actions de l'utilisateur (feedback visuel de l'appui sur un bouton ou transition de vue). De même, on veut des transitions cohérentes, qui donnent le même genre de feedback pour les mêmes types d'actions.

Ces quelques principes-clés esquissés, on montrera comment Qt Commercial peut être utilisé pour rendre l'essence du style Metro par le biais de deux applications simples : une vue en grille avec QML et une application de contrôle de pivot en C++.

3. Créer une vue en grille avec QML et un style Metro

Cet exemple de vue en grille montre comment les composants QML peuvent être utilisés pour créer des applications au style Metro, semblables à l'écran de démarrage en tuiles de Windows 8 :

Cet écran de démarrage consiste en un en-tête et une grille en dessous. La vue en grille est alignée sur la gauche et déborde du côté droit. La grille peut être déplacée sur la gauche ou la droite, les applications peuvent être démarrées en touchant les tuiles. Pour suivre les principes de conception édictés dans l'introduction, on définit quelques paramètres de style pour les composants de l'interface :

main.qml
```
property int topBarSize: 144
property int barSize: 120
property int tileMargin: 12
property int tileHeaderFontSize: 11
property int tileDateFontSize: 10
property int appHeaderFontSize: 36
property string appBackground: "#262626"
property string tileBackground: "#86bc24"
property string textColor: "white"
property string uiFont: "Segoe UI"
```

Ces propriétés définissent la structure, les couleurs et les polices pour l'interface. Comme on peut le voir, topBarSize, barSize et tileMargin ont une valeur de douze pixels, des incréments d'autant de pixels pour rendre la disposition cohérente. Le type de police est défini à Segoe UI. Une fois le style défini, on s'occupe de la structure de l'interface. On ne va pas passer en revue tout le code, seulement les éléments importants.

main.qml
```
// Barre supérieure
Rectangle {
    id: topBar
    anchors.left: leftBar.right
    anchors.top: parent.top
    height: topBarSize
    color: appBackground
    Text {
        ...
        text: qsTr("Qt Commercial Blog")
        font.family: uiFont;
        font.pointSize: appHeaderFontSize;
```

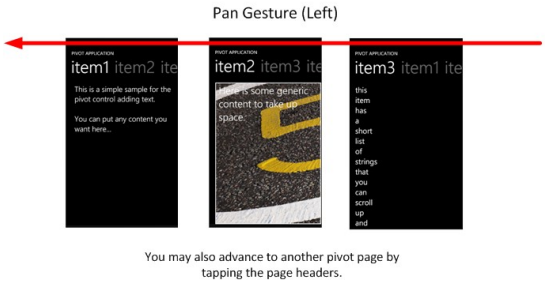
```
        color: textColor
    }
}

// Barre de gauche
Rectangle {
    id: leftBar
    anchors.left: parent.left
    anchors.top: parent.top
    width: barSize
    height: parent.height
    color: appBackground
    ...
}

// Vue en grille
GridView {
    id: grid
    anchors.left: leftBar.right
    anchors.top: topBar.bottom
    flow: GridView.TopToBottom
    width: parent.width - leftBar.width
    height: parent.height - topBar.height -
bottomBar.height
    cellHeight: parseInt(grid.height / 3)
    cellWidth: parseInt(cellHeight * 1.5)
    clip: false
    model: feedModel
    delegate: RssDelegate {}
    ...
}

// Barre du bas
Rectangle {
    id: bottomBar
    anchors.top: grid.bottom
    anchors.left: leftBar.right
    width: parent.width - leftBar.width
    height: barSize
    color: appBackground
    ...
}
```

La structure est assez simple : elle est constituée d'une barre supérieure, d'une barre à gauche, d'une barre en bas et de la vue en grille centrée, avec les paramètres de style déjà définis. Le composant GridView (Lien 24) est standard en QML et affiche ses items en grille, de manière horizontale. Les tailles sont définies avec les propriétés cellHeight et cellWidth. La rognure est désactivée, on peut donc déplacer la grille vers la gauche ou la droite "sur" l'écran. Cette vue supporte aussi le déplacement cinétique avec une courbe de lissage appropriée par défaut. Le contenu de la vue est fourni par le flux RSS Qt Commercial (Lien 25), récupéré par XmlListModel (Lien 26).

C'est tout. L'application QML en grille est stylée à la mode Metro. On peut en vérifier le résultat dans la démonstration en fin d'article.

4. Créer un contrôle de pivot avec Qt en C++

Dans ce second exemple, on va montrer comment implémenter un contrôle de pivot avec un style Metro à l'aide de Qt en C++. Il s'agit d'une version Metro de l'habituel contrôle d'onglets avec quelques exceptions. On peut définir plusieurs pages pour la même fenêtre ; chaque page consiste en un certain type d'informations ou d'autres contrôles. Les items de l'en-tête du pivot glissent sur l'écran vers la droite et l'item sélectionné est toujours déplacé sur la gauche de l'écran.



Pour les briques de base de l'application, on a décidé d'utiliser les classes QGraphicsView et QgraphicsTextItem (Lien 27). Comme dans l'exemple précédent, on a défini quelques paramètres pour l'interface.

Style.h

```
const int componentMargin = 24;
const int bodyTextSize = 24;
const int headerTextSize = 48;
const QFont headerFont = QFont("Segoe UI",
headerTextSize);
const QFont bodyFont = QFont("Segoe UI",
bodyTextSize);
const QColor uiTextColor = Qt::white;
const QString backgroundStyle = "background-
color: rgba(26,26,26)";
const int animationTime = 400 ;
```

componentMargin définit une marge cohérente pour tous les composants de l'interface, tant horizontalement que verticalement. La police sera Segoe UI. Le bout de code ci-après montre la création des items de l'en-tête du pivot.

tabview.cpp

```
...
const int itemCount = 6;
...
QGraphicsTextItem *tmp;
...
for(int i = 0; i < itemCount; ++i) {
    tmp = new QGraphicsTextItem();
    text = "loremIpsum";
    text = text.append(QString("%1").arg(i+1));
    tmp->setPlainText(text);
    tmp->setFont(headerFont);
    tmp->adjustSize();
    tmp->setDefaultTextColor(uiTextColor);
    // sous le texte de l'en-tête
    tmp->setPos(xPos,(componentMargin * 2 +
bodyTextSize));
    // calculer la position de l'item suivant :
c'est la somme de la position actuelle,
        // de la largeur de l'item et de la
marge.
    xPos = xPos + tmp->textWidth() +
componentMargin;
    ...
    mHeaderItems.append(tmp);
```

```
    scene()->addItem(tmp);
}
```

Six éléments de l'en-tête du contrôle sont créés. Pour chaque en-tête, on doit définir du contenu. Dans ce cas, le contenu sera du texte, on utilise donc QGraphicsTextItem pour l'afficher. Puisque QGraphicsTextItem est utilisé dans l'en-tête et les items du contenu, la création de ces items sera semblable à l'instanciation de l'en-tête et de son contenu. Les seules différences seront la police (bodyFont) et la position (déterminée par rapport aux items de l'en-tête).

Pour apporter du mouvement aux interfaces par le biais d'animations, on utilise la classe QpropertyAnimation : Lien 28. Le bout de code suivant montre la création de l'item d'animation.

tabview.cpp

```
...
QPropertyAnimation *anim;
anim = new QPropertyAnimation(tmp, "pos");
anim->setDuration(animationTime);
anim->setPropertyName("pos");
anim->setEasingCurve(QEasingCurve::OutCirc);
mGroupAnimHeader->addAnimation(anim);
mHeaderAnimations.append(anim);
...
mHeaderItems.append(tmp);
scene()->addItem(tmp) ;
```

Le style d'interface Metro recommande l'utilisation de deux types de courbes d'assouplissement : logarithmique pour l'entrée à l'écran et exponentielle pour la sortie. Dans ce cas, QEasingCurve::OutCirc est utilisé pour les deux : Lien 29. Ce type de courbe est fort semblable à une courbe logarithmique pour l'animation. Le bout de code suivant montre l'utilisation des animations.

tabview.cpp

```
void TabView::mouseMoveEvent(QMouseEvent *event)
{
    int xDif = event->pos().x() -
mMouseXPosition;
    ...
    if(xDif < 0) {
        ...
        startContentAnimation(ESweepLeft);
        startHeaderAnimation(ESweepLeft);
    } else if(xDif > 0) {
        ...
        startContentAnimation(ESweepRight);
        startHeaderAnimation(ESweepRight);
    }
    mMouseXPosition = event->pos().x();
}

void TabView::startContentAnimation(int
direction)
{
    QPropertyAnimation *anim;
    QGraphicsTextItem *text;
    // init animation items
    for(int i = 0; i < itemCount; ++i) {
        text = mContentItems.at(i);
        anim = mContentAnimations.at(i);
        QPointF start = text->pos();
        QPointF end = start;
        if(direction == ESweepLeft)
            end.setX( end.x() - text->textWidth()
- componentMargin);
        else
            end.setX( end.x() + text->textWidth()
+ componentMargin);
        anim->setStartValue(start);
        anim->setEndValue(end);
    }
    mGroupAnimContent->start();
}
```

L'animation est démarrée pour le pivot et les items du contenu quand l'écran est balayé vers la gauche ou la droite. Ce mouvement est détecté dans la fonction mouseMoveEvent et chaque fois qu'une nouvelle animation débute ou se termine, les points étant calculés à l'aide de sa direction. L'opacité des items est définie à 0.6 en cas de non-mise en évidence. Pour les éléments mis en évidence, l'opacité est de 1.0 avec utilisation d'une courbe d'assouplissement linéaire pendant les animations.

En résumé, on a maintenant un contrôle de pivot construit à l'aide de la classe QGraphicsTextItem avec une transition entre les vues avec la classe QPropertyAnimation.

5. Vidéo et code

Le résultat de ces péripéties est résumé dans cette vidéo : Lien 30. Le code est d'ailleurs disponible : Lien 31.

Retrouvez l'article de Sami Makkonen traduit par Thibaut Cuvelier en ligne : Lien 32

Commencer facilement avec Boost Graph

Boost Graph (BGL) est une bibliothèque permettant de créer et manipuler des graphes. Ce tutoriel est une introduction pratique à BGL permettant de prendre en main facilement les fonctionnalités de base.

1. Que propose Boost Graph ?

Boost Graph (BGL) propose une interface standard pour manipuler des graphes. Cette interface est similaire aux conteneurs de la bibliothèque standard et permet l'accès aux différents éléments à l'aide d'itérateurs. Il est donc possible d'utiliser les algorithmes de la bibliothèque standard pour travailler sur les graphes ou d'utiliser les algorithmes spécifiques pour les graphes fournis par Boost Graph. On peut aussi manipuler les éléments du graphe, accessibles via la classe de traits : les sommets (ou nœuds, vertex dans BGL), les arcs (pour un graphe orienté ou arêtes pour un graphe non orienté, edge dans BGL) et le graphe lui-même (graph dans BGL)

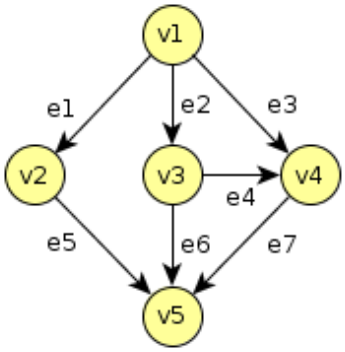


Figure 1: Exemple de graphe avec 5 sommets (v1 à v5) et 7 arcs (e1 à e7) qui sera utilisé tout au long de ce tutoriel

2. Comment débuter avec Boost Graph ?

2.1. Créer un graphe générique

Boost Graph propose de nombreux outils qui nécessitent un certain temps pour tout appréhender. Cependant, il est possible d'utiliser la classe adjacent_list comme boîte à outils généraliste.

```
#include <boost/graph/adjacency_list.hpp>
```

Les informations relatives à chaque élément d'un graphe sont enregistrées dans des structures :

```
struct VertexProperties { ... };
struct EdgeProperties { ... };
struct GraphProperties { ... };
```

On peut alors créer un graphe générique basé sur adjacent_list. Pour des raisons pratiques, il est préférable de créer un typedef sur cette structure puisqu'elle sera réutilisée plusieurs fois :

Définition du graphe
```
typedef boost::adjacency_list<
   boost::vecS, boost::vecS,
boost::bidirectionalS,
   boost::property<boost::vertex_bundle_t,
VertexProperties>,
   boost::property<boost::edge_bundle_t,
EdgeProperties>,
   boost::property<boost::graph_bundle_t,
GraphProperties>,
> Graph;
```

Création du graphe
```
Graph  g;
```

2.2. Comment manipuler les sommets ?

Le type correspondant à la description d'un sommet est accessible via la classe de traits graph_traits :

```
typedef
boost::graph_traits<Graph>::vertex_descriptor
vertex_t;
```

La fonction add_vertex permet d'ajouter un sommet dans un graphe. Les informations attachées au sommet peuvent être spécifiées lors de la création :

```
struct VertexProperties
{
   std::string name;
   unsigned id;
   VertexProperties() : name(""), id(0) {}
   VertexProperties(std::string const& n,
unsigned i) : name(n), id(i) {}
};
```

Appel du constructeur par défaut
```
vertex_t v1 = boost::add_vertex(g);
```

Appel du constructeur avec paramètres
```
vertex_t v2 =
boost::add_vertex(VertexProperties("toto", 12),
g);
```

L'opérateur [] permet de récupérer les informations d'un sommet. Il retourne une référence, ce qui permet de pouvoir modifier les informations :

Référence constante
```
VertexProperties const& vertexProperties = g[v1];
std::cout << "Vertex name : " <<
vertexProperties.name << std::endl;
```

Référence non constante
```
VertexProperties& vertexProperties = g[v2];
v2.id = 17;
```

La fonction num_vertices permet de connaître le nombre de sommet dans un graphe :

Nombre de sommets
```
boost::graph_traits<Graph>::vertices_size_type s
= num_vertices(g);
```

2.3. Comment manipuler les arcs ?

Le type correspondant à la description d'un arc est accessible via la classe de traits graph_traits :

```
typedef
boost::graph_traits<Graph>::edge_descriptor
edge_t;
```

La fonction add_edge permet de connecter deux sommets. Les informations attachées à l'arc peuvent être spécifiées lors de la création :

```
struct EdgeProperties
{
   float weight;
   float distance;
   EdgeProperties() : weight(0.0), distance(0.0)
{}
   EdgeProperties(float w, float d) : weight(w),
distance(d) {}
};
```

Appel du constructeur par défaut
```
std::pair<Graph::edge_descriptor, bool> e1 =
boost::add_edge(v1, v2, g);
```

Appel du constructeur avec paramètres
```
std::pair<Graph::edge_descriptor, bool> e2 =
   boost::add_edge(v1, v2, EdgeProperties(1.0,
50.0), g);
```

L'opérateur [] permet de récupérer les informations d'un arc. Il retourne une référence, ce qui permet de pouvoir modifier les informations :

Référence constante
```
EdgeProperties const& edgeProperties = g[e1];
std::cout << "Edge weight : " <<  edgeProperties.
weight << std::endl;
```

Référence non constante
```
EdgeProperties& edgeProperties = g[e2];
e2. distance = 103.8;
```

La fonction num_edges permet de connaître le nombre de sommets dans un graphe :

Nombre d'arcs
```
boost::graph_traits<Graph>::edges_size_type s =
num_edges(g);
```

Il est possible de récupérer les descripteurs des sommets associés à un arc à l'aide des fonctions source() et target() :

```
Graph::target_descriptor v1 = boost::target(e1,
g);
Graph::target_descriptor v2 = boost::source(e5,
g);
if (v1 == v2)
   std::cout << "Same vertex" << std::endl;
```

2.4. Comment supprimer des sommets et des arcs ?

Boost Graph fournit plusieurs fonctions pour supprimer des éléments d'un graphe. Attention, ces fonctions invalident les itérateurs existants.

Suppression de tous les sommets et arcs d'un graphe
```
clear(g);
```

Suppression des arcs partant ou arrivant à un sommet
```
clear_in_edges(v1, g);
clear_out_edges(v2, g);
```

Suppression de tous les arcs partant et arrivant à un sommet
```
clear_vertex(v1, g);
```

Suppression d'un arc
```
remove_edge(v1, v2, g);
remove_edge(e3, g);
```

Suppression d'un sommet
```
// il est nécessaire de supprimer les arcs liés à
un sommet...
clear_vertex(v1, g);
// ... avant de le supprimer
remove_vertex(v1, g);
```

2.5. Comment utiliser des algorithmes ?

Boost Graph fournit des fonctions pour récupérer des itérateurs sur les sommets ou les arcs d'un graphe. Il est ensuite possible de les utiliser directement dans les algorithmes de la bibliothèque standard.

Récupérer tous les sommets
```
std::pair<vertex_iterator_t, vertex_iterator_t>
it = boost::vertices(g);
```

Récupérer tous les arcs
```
std::pair<edge_iterator_t, edge_iterator_t> it =
boost::edges(g);
```

Récupérer les arcs partant d'un sommet
```
std::pair<out_edge_iterator_t,
out_edge_iterator_t> it = boost::out_edges(v1,
g);
```

Récupérer les arcs arrivant sur un sommet
```
std::pair<in_edge_iterator_t, in_edge_iterator_t>
it = boost::in_edges(v1, g);
```

Récupérer les sommets adjacents à un sommet
```
std::pair<adjacency_iterator_t,
adjacency_iterator_t> it =
    boost::adjacent_vertices(v1, g);
```

On peut alors parcourir tous les éléments sélectionnés. Par exemple, pour afficher le nom de tous les sommets :

```
std::pair<vertex_iterator_t, vertex_iterator_t>
it = boost::vertices(g);
for( ; it.first != it.second; ++it.first)
   std::cout << get(boost::vertex_bundle, g)
[*it.first].name << std::endl;
```

2.6. Quels sont les algorithmes proposés par Boost Graph ?

- Breadth First Search
- Connected Components
- Depth First Search
- Strongly Connected Components
- Uniform Cost Search
- Dynamic Connected Components
- Dijkstra's Shortest Paths
- Topological Sort
- Bellman-Ford Shortest Paths
- Transpose
- Johnson's All-Pairs Shortest Paths
- Reverse Cuthill Mckee Ordering
- Kruskal's Minimum Spanning Tree
- Smallest Last Vertex Ordering
- Prim's Minimum Spanning Tree
- Sequential Vertex Coloring

Voir la documentation de Boost Graph pour l'utilisation de ces algorithmes.

3. Aller un peu plus loin avec Boost Graph

3.1. Utiliser la fonction tie()

La fonction tie de boost/tuple/tuple.hpp permet de récupérer une std::pair directement dans deux variables :

```
Graph::edge_descriptor e1;
bool succes;
tie(e1, succes) = boost::add_edge(v1, v2, g);
if (succes)
   // using edge e1
```

3.2. Que signifient les paramètres passés à adjacent_list ?

adjacency_list est une classe template acceptant plusieurs paramètres permettant de spécifier le comportement de cette classe. Ces paramètres permettent de choisir les types de conteneurs utilisés en interne (OutEdgeList, VertexList et EdgeList), les propriétés associées aux éléments du graphe (VertexProperties, EdgeProperties, GraphProperties et Directed).

Voici la liste des paramètres et les valeurs par défaut de adjacency_list :

```
adjacency_list<
   OutEdgeList          = vecS,
   VertexList           = vecS,
   Directed             = directedS,
   VertexProperties     = no_property,
   EdgeProperties       = no_property,
   GraphProperties      = no_property,
   EdgeList             = listS
>
```

3.3. Les types des conteneurs

Les types de conteneurs (OutEdgeList, VertexList et EdgeList) peuvent être choisir en utilisant les tags précisés dans la liste suivante :

```
vecS            = std::vector
listS           = std::list
slistS          = std::slist
setS            = std::set
multisetS       = std::multiset
hash_setS       = std::hash_set
```

Le choix du type de conteneur aura une influence sur les performances des algorithmes utilisés. La complexité des fonctions de Boost Graph en fonction du type de conteneur est indiquée dans la page suivante : using_adjacency_list (Lien 33).

3.4. L'orientation des arcs

Le paramètre Directed permet de préciser si le graphe est orienté ou non et si les arcs seront unidirectionnels ou bidirectionnels.

```
undirectedS             = graphe non orienté
directedS               = graphe orienté avec des
arcs unidirectionnels
bidirectionalS          = graphe orienté avec des
arcs bidirectionnels
```

En fonction du type de graphe, certaines fonctions ne seront pas disponibles. Par exemple, avec un graphe orienté unidirectionnel, la fonction in_edges, permettant de récupérer la liste des arcs entrants, n'est pas utilisable. De même, certains algorithmes ne seront possibles que pour certains types de graphes.

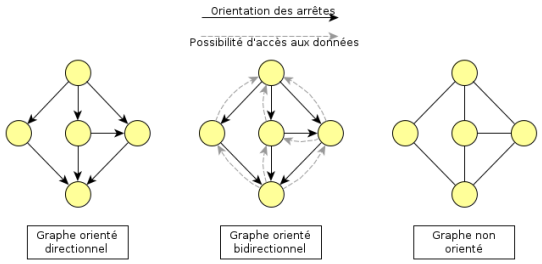


Figure 2 : Différents types de graphes

directedS ne permet pas de récupérer les arcs entrants dans un sommet (in_edges) d'où l'existence de bidirectionalS. Tous deux sont des graphes orientés au niveau conceptuel. Par défaut, on choisit entre undirectedS ou birectionalS en fonction de la nature du graphe (non orienté / orienté). directedS sera une optimisation possible pour consommer moins de mémoire.

3.5. Les propriétés associées

Pour les propriétés associées aux éléments d'un graphe, on peut soit utiliser le mot-clé no_property pour ne pas associer d'informations, soit utiliser la méthode décrite dans les chapitre 3-B pour les sommets et 3-C pour les arcs, soit utiliser une des méthodes décrites ensuite.

Il existe deux approches possibles pour enregistrer les informations dans un graphe : utiliser des conteneurs

externes (« external property storage ») ou directement dans un graphe (« internal properties »).

3.5.1. Enregistrer les informations dans des conteneurs externes

3.5.1.1. Créer une property_map

Il existe plusieurs classes dans property_map mais nous allons décrire uniquement associative_property_map ici. Une property_map prend en paramètre un conteneur associatif. Nous utilisons pour Boost Graph un descripteur de sommets ou d'arcs comme clé :

```
typedef std::map<vertex_t, std::string>
names_property_t;
names_property_t names;
boost::associative_property_map<names_property_t>
names_map(names);

typedef std::map<edge_t, float>
weights_property_t;
weights_property_t weights;
boost::associative_property_map<weights_property_
t> weights_map(weights);
```

3.5.1.2. Accéder aux informations

Pour accéder aux données, on utilise les fonctions put(), get() et l'opérateur [] :

Avec les fonctions put et get

```
std::string v1_name = get(names_map, v1) ;
put(names_map, v2, « toto »);
```

Plus directement, avec l'opérateur []

```
std::string v1_name = names_map[v1];
names_map[v2] = « toto »;
```

3.5.2. Enregistrer les informations directement dans le graphe

Il est possible d'utiliser les paramètres template VertexProperties, EdgeProperties et GraphProperties de adjacent_list pour enregistrer des informations directement dans le graphe. Chaque information est identifiée par un tag, auquel on spécifie un type de données et une valeur.

Par exemple, on peut associer un nom à chaque sommet :

```
// typedef
typedef boost::property<boost::vertex_name_t,
std::string> VertexProperties;

// création d'un sommet
vertex_t v1 = boost::add_vertex(« toto », g);

// récupérer le nom
std::cout << « Vertex name : " <<  g[v1] <<
std::endl;

// modifier le nom
v1 = « titi »;
```

3.5.2.1. Les tags prédéfinis

Il existe un certain nombre de tags prédéfinis :

- vertex_index_t ;
- edge_residual_capacity_t ;
- vertex_isomorphism_t ;
- vertex_index1_t ;
- edge_reverse_t ;
- vertex_invariant_t ;
- vertex_index2_t ;
- vertex_distance_t ;
- vertex_invariant1_t ;
- edge_index_t ;
- vertex_root_t ;
- vertex_invariant2_t ;
- graph_name_t ;
- vertex_all_t ;
- vertex_degree_t ;
- vertex_name_t ;
- edge_all_t ;
- vertex_out_degree_t ;
- edge_name_t ;
- graph_all_t ;
- vertex_in_degree_t ;
- edge_weight_t ;
- vertex_color_t
- vertex_discover_time_t
- edge_weight2_t ;
- vertex_rank_t ;
- vertex_finish_time_t ;
- edge_capacity_t ;
- vertex_predecessor_t.

3.5.2.2. Créer ses propres tags

On peut également créer ses propres tags avec le code suivant :

```
namespace boost {
    enum vertex_data_t { vertex_data };
    BOOST_INSTALL_PROPERTY(vertex, data);
}
typedef property<data_t, int> VertexProperties;
```

3.5.2.3. Enregistrer plusieurs informations par élément du graphe

Boost property prend un troisième paramètre template, permettant de chaîner plusieurs propriétés :

```
boost::property <boost::vertex_name_t,
std::string,                        // nom
    boost::property<boost::vertex_index1_t,
unsigned,           // id
        boost::property<boost::vertex_distance_t,
float> > >    // distance
```

Dans ce cas, il est nécessaire d'utiliser les fonctions put() et get() en indiquant le tag utilisé pour récupérer la property_map correspondant au tag :

```
// modifier le nom
get(boost::vertex_name_t, g)[v1] = « titi »;
```

```
// afficher le nom
std::cout << « Vertex name : " <<
get(boost::vertex_name_t, g)[v1] << std::endl;
```

3.6. Représentation interne et adjacent_matrix

La classe adjacent_list utilise en interne une liste de sommets auxquels sont associés une liste d'arcs. Il existe une autre structure, adjacent_matrix, qui utilise un tableau 2D pour représenter chaque arc, ce qui permet un accès en O(1) aux arcs.

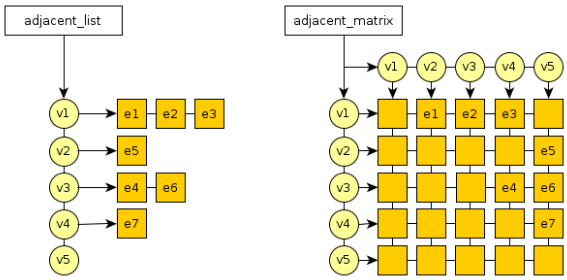


Figure 3: Représentation interne de adjacent_list et adjacent_matrix

4. Références

- Lien 34 ;
- Lien 35 ;
- Boost Graph Library, The: User Guide and Reference Manual de Jeremy G. Siek, Lie-Quan Lee et Andrew Lumsdaine.

Retrouvez l'article de Guillaume Belz en ligne : Lien 36

Présentation de Java SE 7

La demoiselle s'est fait attendre ! C'est le moins que l'on puisse dire. Alors que l'on était jusque-là habitué à avoir une nouvelle version tous les deux ans en moyenne, il aura fallu quatre ans et demi pour voir débarquer Java SE 7. Mais il est vrai que son développement fut assez mouvementé, entre les ajouts/suppressions de fonctionnalités, les divers reports et surtout le rachat de Sun par Oracle...

1. JSR 334 - Petites améliorations du langage (le projet « Coin »)

L'objectif du projet « Coin » est d'apporter de petites modifications au langage. On est loin ici des « révolutions » de **Java 5.0** (Generics, enums, annotations...). L'idée principale étant d'apporter de petits correctifs ou évolutions simples et efficaces pour faciliter la vie du développeur.

1.1. Expression littérale binaire

En Java, comme dans un grand nombre de langages, les expressions numériques entières peuvent être représentées de trois manières distinctes :

- le plus couramment, en notation décimale (exemple : **255**) ;
- en notation hexadécimale, préfixée par **0x** (exemple : **0xff**) ;
- en notation octale, préfixée par **0** (exemple : **0377**).

Désormais, le langage s'enrichit avec la notation binaire, préfixée par **0b**, qui permet donc de représenter un nombre directement sous sa forme binaire (exemple : **0b11111111**).

Quelques exemples :

```
// Un 'byte' (sur 8 bits)
byte aByte = (byte)0b00100001;

// Un 'short' (sur 16 bits)
short aShort = (short)0b1010000101000101;

// Quelques 'int' (sur 32 bits) :
int anInt1 = 0b10100001010001011010000101000101;
int anInt2 = 0b101; // Les zéros inutiles peuvent être ignorés
int anInt3 = 0B101; // Le B peut être en minuscule/majuscule

// Un 'long' (sur 64-bit)
long aLong = 0b1010000101000101101000010100010110101000010100010110101000010100L;
```

Le principal intérêt étant de simplifier la manipulation de valeurs binaires, en permettant de les représenter directement sous cette forme, ce qui facilite la lecture du code pour la manipulation des bits.

1.2. Formatage des expressions numériques

Comme dans la majorité des langages, les expressions numériques sont représentées par une suite de chiffres sans aucun séparateur, si ce n'est le point qui sépare la partie entière de la partie décimale pour les valeurs flottantes.

Toutefois, lorsqu'on manipule des expressions numériques représentant des nombres assez grands, on peut vite se retrouver avec des problèmes de lisibilité. Il faut alors « compter » le nombre de chiffres présents dans l'expression. **Java 7** apporte une solution à ce problème, en permettant d'utiliser le caractère underscore (_) au milieu de n'importe quelle expression numérique afin de séparer des groupes de chiffres de manière totalement arbitraire, afin de rendre le tout plus lisible :

Quelques exemples :

```
double oneMilliard = 1__000_000_000.000_000;
long creditCardNumber = 1111_2222_3333_4444L;
long hexBytes = 0xFF_EC_DE_5E;
long hexWords = 0xCAFE___BABE;
long bytes = 0b11010010_01101001_10010100_10010010;
```

Le seul et unique intérêt consiste à améliorer la relecture du code. La présence des underscores dans une expression numérique n'a bien sûr strictement aucun impact sur le code généré ni sur la valeur de l'expression.

Les underscores doivent impérativement être situés entre deux caractères numériques. Ils ne peuvent donc pas être utilisés au début ou à la fin de l'expression, ni autour du séparateur de décimales pour les nombres réels.
Exemples de valeurs incorrectes : **3_.14**, **_1000**, **10_**, **0x_1**...

1.3. Switch sur les String

Fini les multiples **if/else** pour comparer la valeur d'une chaine ! Il est enfin possible d'utiliser un **switch** sur une **String**, afin de comparer une chaîne de caractères avec des valeurs constantes (définies dès la compilation) :

Exemple d'utilisation du switch(String) :

```
String action = ...

switch(action) {
case « save »:
    save();
    break;
case « save-as »:
    saveAs();
    break;
case « update »:
```

```
    update();
    break;
case "delete":
    delete();
    break;
case "delete-all":
    deleteAll();
    break;
default:
    unknownAction(action);
}
```

Tout comme le **switch(enum)**, le **switch(String)** n'accepte pas de valeur **null** (cela provoquera une exception).

En plus d'une meilleure lisibilité, cela permet également au compilateur de générer un code bien plus efficace qu'une succession de **if/else**. Le **switch(String)** se décompose en réalité en deux **switch**es. Le premier se base sur le **hashCode()** afin de séparer les différentes valeurs plus facilement, avant de vérifier l'égalité via **equals()**. Ceci permettra de déterminer l'index du second **switch**, qui reprend le même format que le **switch(String)** original, mais en utilisant des valeurs numériques (ce qui permet de conserver le même ordre des **case** dans tous les cas).

Ainsi le code ci-dessus est à peu près équivalent au code suivant :

Pseudo-code équivalent au switch(String) :

```
String action = ...

int switchIndex = -1;

// On découpe
switch (action.hashCode()) {
case 3522941: // « save ».hashCode()
    if (« save ».equals(action))
        switchIndex = 1;
    break;
case 1872766594: // « save-as ».hashCode()
    if (« save-as ».equals(action))
        switchIndex = 2;
    break;
case -838846263: // « update ».hashCode()
    if (« update ».equals(action))
        switchIndex = 3;
    break;
case -1335458389: // "delete".hashCode()
    if ("delete".equals(action))
        switchIndex = 4;
    break;
case 1763419775: // "delete-all".hashCode()
    if ("delete-all".equals(action))
        switchIndex = 5;
    break;
}

switch (switchIndex) {
case 1: // « save »
    save();
    break;
case 2: // « save-as »
    saveAs();
    break;
case 3: // « update »
    update();
    break;
case 4: // « delete »
    delete();
    break;
case 5: // "delete-all"
    deleteAll();
    break;
default:
    unknownAction(action);
}
```

Lorsque les différentes chaînes du **switch** disposent de **hashCode()** différents, l'accès se fera donc en temps constant quelle que soit la valeur recherchée, contrairement aux multiples **if/else** où le temps dépend de l'emplacement dans la liste.

Dans le cas de deux chaînes du **switch(String)** (ce qui est peu probable dans un cas réel), le premier **switch** utilisera une succession de **if/else** pour les distinguer. Par exemple les chaînes « bb » et « cC » possèdent le même **hashCode()**. Lorsqu'on les utilise dans un **switch(String)**, cela générera dans le premier **switch** un seul **case** gérant les deux valeurs :

Pseudo-code partiel en cas de conflit de hashCode :

```
case 3136: // « bb ».hashCode() OU
« cC ».hashCode();
    if ("bb".equals(action))
        switchIndex = 3;    // selon l'index du
case dans le switch original
    else if (« cC ».equals(action))
        switchIndex = 7;    // selon l'index du
case dans le switch original
    break;
```

1.4. Type Inference sur la création d'une instance Generics (le losange)

Le « **Type Inference** » est une notion apparue avec les **Generics** de **Java 5.0**. Elle permet au compilateur de déterminer le type du paramétrage **Generics** selon le contexte, afin d'éviter d'avoir à le préciser explicitement dans le code.

Dans **Java 7** le « **Type Inference** » a été amélioré afin de prendre en compte le cas particulier de la création d'une instance **Generics**. En effet, la plupart du temps une création d'instance est affectée à une référence dont le type est déclaré avec le même paramétrage **Generics**. Cela aboutit à une répétition qui peut s'avérer assez lourde dans certains cas :

Quelques exemples avec Java 5/6 :

```
List<String> list = new ArrayList<String>();

Map<Reference<Object>,Map<String,List<Object>>>
map = new
HashMap<Reference<Object>,Map<String,List<Object>
>>();
```

Avec l'amélioration du « **Type Inference** » de **Java 7**, le compilateur pourra donc retrouver implicitement le paramétrage **Generics** de la création de l'instance, selon celui utilisé dans la variable à laquelle on l'affectera.

Les mêmes exemples avec Java 7 :

```
List<String> list = new ArrayList<>();

Map<Reference<Object>,Map<String,List<Object>>>
map = new HashMap<>();
```

Le paramétrage **Generics** du constructeur est simplement omis, si bien qu'il ne reste plus que les crochets <>, d'où le surnom de « syntaxe en losange ».
Cela permet d'éviter de saisir deux fois la liste des paramètres **Generics** (une fois à la déclaration et une fois lors de la construction de l'instance).

Retrouvez la suite de l'article de Fred Martini en ligne : Lien 37

Interview d'Hugo Lassiège, du comité de sélection des présentations pour Devoxx France

À l'occasion de Devoxx France, dont l'édition 2012 se déroulera pour la première fois à Paris, Hugo Lassiège a volontiers accepté de présenter l'organisation de la conférence et plus particulièrement à propos du "comité de sélection des orateurs" dont il fait partie.

1. Introduction

Hugo Lassiège, notre confrère et rédacteur de nombreux articles pour Developpez.com vient de prendre part à l'aventure de l'année, dans le petit monde du développement. En effet, Hugo a rejoint l'équipe organisatrice de Devoxx France et en particulier le comité qui sélectionne les présentations qui nous seront proposées.

Hugo Lassiège

Dans cette interview, Hugo revient sur son parcours puis nous présente sa vision de Devoxx. Il nous décrit les coulisses de l'événement, notamment les critères de sélection des sujets proposés par les orateurs.

2. Présentation d'Hugo

En quelques mots, qui es-tu ?

Pour être très succinct, je suis développeur et je suis fier de l'être malgré la trentaine passée.

Par contre je ne suis pas chauve. Si vous ne voyez pas le rapport, je vous conseille l'excellent billet de Nicolas Martignole : « Développeur après 31 ans ? Ridé et chauve tu seras. » (Lien 38)

J'anime un petit blog si jamais vous voulez en savoir plus : Lien 39.

Quel est ton parcours ?

Là forcément je fais un peu plus long car j'ai eu la chance d'avoir plusieurs expériences assez différentes. Je suis passé par le service où j'ai fait mes premières armes. Ensuite j'ai pu intégrer un éditeur de logiciel chez lequel je me suis bien éclaté pendant quelques années. J'ai ensuite enchainé comme freelance. C'est d'ailleurs ce qui m'a permis de rencontrer Nicolas Martignole, José Paumard et Antonio Goncalves au sein des Zindeps : Lien 40. Et plus récemment je viens de fonder localizeyourapps.com (Lien 41) et lateral-thoughts.com .(Lien 42) Je donne aussi quelques cours à l'IUT.

Qu'est-ce qui te motive dans la vie, dans le travail ?

En deux mots : apprendre, rencontrer.

J'aime les challenges intellectuels, voir de nouvelles technos et rencontrer des gens. Dès que j'ai senti que je n'apprenais plus dans un boulot ou que je me suis senti stagner dans un environnement, j'ai fait en sorte de changer.

Côté techno, j'ai eu l'opportunité de bosser en C, C++, C#, Java, PHP, PL/SQL, shell et j'ai hâte d'en voir plus.

Ma préférence en ce moment va vers l'écosystème Java mais je me considère comme pragmatique, il faut faire ses choix en fonction de ses besoins.

Un de mes « dadas », c'est aussi ce qui tourne autour de la qualité, l'intégration continue et le déploiement continu.

Quel est ton rôle au sein de Developpez.com ?

Je suis rédacteur sur Developpez.com. C'est amusant d'ailleurs que l'on continue de me contacter pour de vieux articles, par exemple certains que j'ai écrits en 2005/2006 et sur lesquels j'ai beaucoup de mal à apporter mon support puisqu'ils sont devenus obsolètes.

Developpez.com m'a beaucoup apporté quand j'ai débuté car la communauté est très active. J'avoue qu'aujourd'hui je suis moi-même un peu moins impliqué que j'ai pu l'être.

Écrire des articles m'a poussé à approfondir un peu plus certains sujets.

Combien de temps passes-tu sur Developpez.com ?

Le thread d'actualités de Developpez.com fait partie des flux que je lis quotidiennement pour faire de la veille techno. Par contre je passe assez peu sur le forum. J'utilise surtout les ressources en lignes, tutoriels et cours.

Auprès de ton entourage professionnel, quand il y a une question technique, le forum Developpez.com est-il

un réflexe ?

Parfois mais ce n'est pas systématique. La majorité du temps, les questions que l'on se pose ont déjà une réponse, on est rarement le premier à rencontrer un problème. Le premier réflexe c'est donc la doc officielle, le bug tracker, le wiki ou toute autre doc reliée à l'outil dont il est question.

Par contre, plus que le forum, les ressources en ligne de Developpez.com sont assez riches.

Mais sûr, une recherche Google tend à renvoyer plus souvent les résultats de Stack Overflow. C'est peut-être dû au système de vote qui permet d'élire la meilleure réponse. Ce serait d'ailleurs une bonne évolution à considérer pour Developpez.com afin de sélectionner plus rapidement la réponse parmi plusieurs pages de discussion.

3. Devoxx France 2012

Présente-nous ta vision de Devoxx France.

Devoxx France c'est avant tout une grosse conférence de geeks. Le côté francophone fait partie des points positifs de l'événement car il y a des tas de talents francophones et on ne le voit pas assez souvent.

Mais Devoxx ce n'est pas qu'une conférence Java, c'est-à-dire qu'on ne va pas que parler de la sérialisation, des collections ou des API nio2. Devoxx France ce sera aussi l'occasion de voir les technologies de demain, les nouveaux langages sur la JVM ou les pratiques d'entreprise.

Devoxx c'est surtout plusieurs formats très différents qui permettront de mettre les mains dans le cambouis, de voir des retours d'expériences, d'avoir des présentations éclair sur des outils, d'apprendre ou de discuter sur des sujets d'aujourd'hui.

Devoxx France 2012

Que va-t-on voir/entendre durant ces trois jours ?

On va voir une conférence capable d'ouvrir l'esprit sur les technologies de demain, d'aider à mieux comprendre celles d'aujourd'hui et changer sa vision ou améliorer son quotidien lorsqu'on bosse sur celles d'hier.

On va assister à des séances université assez poussées, participer à des ateliers de 3 heures sur des thèmes pointus, découvrir des outils ou des pratiques sur des formats de 15 à 30 minutes et découvrir des tas de speakers intéressants sur des conférences de 1 heure.

Peut-on s'attendre à rencontrer des rock stars ?

Clairement oui. Parmi les rock stars déjà annoncées (Lien 43), on a des membres de Google, du W3C, de Redhat, JRebel, Heroku pour n'en citer que quelques-uns. Et il n'y a que seize personnes annoncées sur la centaine de speakers qui seront présents. Peu de chance que vous déambuliez dans les couloirs sans croiser quelques « rock stars ».

Comment se positionne Developpez.com vis-à-vis de cet événement ?

Developpez.com est partenaire presse. Au-delà de ce partenariat, il y a plusieurs membres de Developpez.com qui ont soumis des présentations à la conférence.

Question organisation, y aura-t-il un vestiaire où déposer nos sacs, manteaux, etc. ?

Oui, une salle est prévue à cet effet.

Y a-t-il une station de Vélib' à côté ?

Il y en a trois dans les environs, dur d'échapper au Vélib' à Paris.

Y aura-t-il un accès WiFi ou 3G ?

Le réseau sera fourni par câble aux orateurs. Une salle de Hands. On aura le WiFi pour son atelier. Avoir du WiFi pour autant de monde est une grosse galère et même Devoxx Belgique n'a pas encore réussi à l'offrir. Pour ce qui est de la 3G, il est possible de la capter dans les salles de conférence mais ce sera une autre paire de manches quand il y aura 800 personnes dans les locaux. Il ne faut pas s'attendre à un réseau très accessible.

Plusieurs de nos membres de province nous ont demandé si nous connaissions des hôtels bien, et pas trop chers, à proximité. Peux-tu nous en conseiller quelques-uns ? Avec des prix négociés ?...

Nous avons un accord avec le Marriot. Cependant cela reste un hôtel 4 étoiles donc pas forcément accessible à toutes les bourses. Nous n'avons pas d'autre accord. À ce propos je conseille de regarder rapidement sur Internet car ce sera aussi la période des vacances. N'hésitez pas à vous éloigner de quelques kilomètres, les transports en commun ne sont si mal que ça à Paris.

4. Comité de sélection des présentations

En quoi consiste le comité de sélection, dont tu fais partie ?

C'est le comité de sélection qui prépare le contenu de la conférence. Le comité de sélection est constitué de onze personnes. Chaque membre du comité a pour objectif de visualiser toutes les candidatures déposées sur le site du cfp.devoxx.com (Lien 44) et de sélectionner celles qui seront au programme de Devoxx France.

Nous pouvons faire des commentaires aux speakers lorsque nous avons besoin de plus de précisions, si le format nous semble non adapté ou parfois pour suggérer des modifications.

Devoxx France 2012

Comment t'es-tu retrouvé embarqué dans l'équipe ?

Je fais partie des Zindeps, un groupe de freelance dont sont membres trois des organisateurs du Paris JUG qui sont aussi à l'origine de Devoxx France. Quand la proposition m'a été faite, j'ai été super emballé par l'idée.

Combien de propositions avez-vous reçues ?

Au moment où j'écris, environ trois cents présentations ont été soumises. On va sûrement devoir ouvrir une salle supplémentaire. On devrait avoir cinq salles en parallèle au lieu de quatre initialement prévues.

Quels sont les critères de sélection que vous utilisez ?

Il y a beaucoup de critères qui rentrent en ligne de compte. Je vais tenter d'en lister quelques-uns :

- la langue : nous avons un objectif de 75 % de présentations francophones ;
- les références : si on peut avoir des retours sur des conférences précédentes c'est plus simple. Les novices ne sont pas écartés pour autant mais on privilégie l'expérience sur les formats longs car parler devant une salle de deux cents personnes n'est pas donné à tout le monde. Dans le même ordre d'idée, on privilégie une personne qui a une légitimité indéniable sur un sujet. Par exemple si le créateur d'un framework fait une proposition il sera mieux noté qu'une personne ayant proposé un sujet sur le même thème ;
- le thème : il faut tenter de coller aux sujets du moment, ceux qui préfigurent l'avenir mais aussi ceux qui nous torturent lorsqu'on a la malchance de tomber sur des projets legacy. Les onze membres du comité ayant des sensibilités différentes, cela permet d'être assez éclectique sur les sujets. Il faut aussi respecter un équilibre. Si par exemple on a vingt conférences sur un sujet, même bien notées, nous devons en écarter pour éviter une surabondance sur ce thème ;
- le résumé du sujet : là-dessus, j'ai envie de dire que les anglophones sont supérieurs avec leurs résumés qui sont plus accrocheurs. Il faut que le résumé réussisse à sortir du lot pour se démarquer des trois cents autres. On voit malheureusement pas mal de résumés assez succincts alors même que le titre laissait présager quelque chose de bien.

Parle-nous du processus de sélection des présentations, depuis la proposition initiale jusqu'à la programmation dans l'agenda.

Le sujet est tout d'abord proposé sur le site du CFP (Call For Paper). Les onze membres du comité ont l'occasion de donner un avis, poser des questions et de noter la présentation.

Si la moyenne des notes est très bonne et que le consensus général est positif, la conférence passe en statut approuvé. Un e-mail est envoyé au conférencier afin qu'il accepte les conditions générales de Devoxx France. Une fois validée, la conférence est en statut « à planifier ». La planification est faite par les tracks leaders et le steering committee qui organise les salles en fonction de multiples contraintes (date, format, cohérence des enchainements, etc.).

Il y a des sujets qui reviennent beaucoup. Parmi eux, vous avez par exemple : la mobilité avec Android, HTML5, le Cloud et les PAAS, Devops, NoSQL. La liste n'est pas exhaustive.

À l'inverse certains sujets sont particulièrement boudés : Swing, Java ME entre autres.

Avec tant de bonnes propositions, vous ne pourrez pas prendre tout le monde. Ça va forcément faire des déçus...

Nous en sommes bien conscients. Plus de deux cents talks seront refusés et parmi eux il y a d'excellentes propositions. Je peux dire qu'on se met une pression forte car il n'est pas évident de faire un choix entre deux sujets super prometteurs. Malheureusement c'est la vie, il y aura des déçus. Mais j'espère que ces personnes comprendront que cela ne remet pas en cause la qualité de leur candidature.

4.1. Bilan

Le comité de sélection des talks vient de terminer son travail. Nous avons laissé quelques jours à Hugo pour se reposer, puis nous lui avons posé une nouvelle série de questions, histoire de dresser un premier bilan.

Un bilan rapide a également été fait durant l'épisode 55 (intitulé « la TVA est en sus ») (Lien 45) de l'excellent podcast « les Castcodeurs ».

La sélection est enfin terminée. Combien y aura-t-il d'orateurs ?

Si la page Devoxxians est exacte, il y a 136 orateurs tous formats confondus, quickies, conférences, hands on, bof et universités : Lien 46.

Le choix a été long et difficile. Quelles en ont été les principales étapes ?

À partir de décembre le comité de sélection s'est réuni toutes les trois semaines environ et tous les quinze jours sur la fin des sélections.

L'ensemble des membres devait voter sur l'application du CFP (Call For Paper). Certains membres avaient une responsabilité supplémentaire sur chaque track. Ces tracks leaders étaient en charge de vérifier la proportion de sujets

et de gérer un budget pour les défraiements sur leurs tracks respectives.

Les dernières réunions avaient pour objectif de sélectionner définitivement les talks en prenant en compte tout les critères : proportion anglais-français, répartition par tracks, représentativité des sujets, etc.

Il y a eu de nombreuses propositions, notamment pour les conférences d'une heure, mais relativement peu de places. Forcément ça va faire des déçus... Quels ont été les critères de sélection ? Y a-t-il eu du favoritisme ?

Je ne pense pas qu'il y ait eu du favoritisme. Il est évident que chacun des membres du comité de sélection avait ses sujets de prédilection mais la diversité des membres du comité a permis que l'on évite de favoriser une communauté plutôt qu'une autre. Je pense que les orateurs sélectionnés sont assez hétérogènes et les sujets très divers.

470 propositions, dont 270 pour les conférences avec seulement 49 places disponibles. Seulement 12 % des propositions ont donc été retenues.

Allez-vous expliquer aux « recalés » pourquoi ils n'ont pas été sélectionnés ? Ça leur permettra éventuellement de proposer une conférence mieux ficelée, mieux présentée, l'année prochaine, ou de voir que le sujet n'est pas conforme, etc.

Il n'est pas prévu de recontacter tous les orateurs non sélectionnés. Je ne sais pas si c'est une bonne chose ou non mais nous avons pensé que ce ne serait pas forcément très productif. Sur ce point nous avons eu des retours négatifs. Nous en rediscuterons pour voir comment améliorer cela l'année prochaine.

Il n'y a pas de « call for paper » à Devoxx (l'original belge). Ça change pas mal de choses dans le choix des orateurs.

Il y a un CFP (Call For Paper) mais beaucoup moins long. L'essentiel des conférences est choisi sur une « wish list ». Pour cette année je pense qu'avoir un CFP à Devoxx France était une bonne chose mais nous allons réfléchir à nouveau sur la façon d'organiser la sélection des sujets. Nous ne sommes pas satisfaits de tout le processus. Nous avons prévu de faire une rétrospective. Je ne m'exprime pas trop dessus pour l'instant car il faudra le faire à froid une fois Devoxx passé.

Y a-t-il des thèmes qui seront plus particulièrement abordés ? Comme le Web ou les futures versions de Java, correspondant à la mode.

Nous avons essayé de respecter un équilibre malgré un grand déséquilibre des soumissions.

On peut dire cependant que le Cloud, HTML5, Nosql, Android vont avoir une bonne visibilité. Mais ça n'empêche pas d'avoir aussi des sujets plus traditionnels comme Spring ou Java7.

À propos, on se souvient que les premières éditions de Devoxx (l'original) étaient consacrées à Java. La version parisienne va-t-elle s'ouvrir aux autres langages ?

Sur ce sujet je dirais qu'il est illusoire de penser désormais que l'on ne fait que du Java dans notre métier. Donc non, Devoxx ne sera pas restreint à Java tout simplement car ça ne correspond pas à notre réalité.

Nous aurons l'occasion de faire un tour côté Web avec HTML5 et JavaScript. On va nous parler des nouvelles formes de stockage de données avec NoSql. Nous aborderons aussi la dématérialisation des services et plein d'autres choses encore.

Personnellement je trouve cela plus représentatif de l'ensemble des sujets que chacun est susceptible de rencontrer dans son quotidien. Même si peu ont la chance de les voir tous ensemble. :-)

L'affiche de Devoxx France indique 75 % de présentations en français. Le contrat sera-t-il respecté ?

Oui ce sera respecté. L'avantage pour un anglophone c'est qu'il devrait pouvoir suivre tout Devoxx en anglais car nous avons essayé d'avoir des talks anglais sur chaque créneau horaire.

Un certain nombre de sessions sont réservées pour les partenaires de Devoxx. Peux-tu nous expliquer de quoi il s'agit ?

Devoxx France repose sur deux sources de financement, les places et les sponsors. Il y a différents niveaux de sponsorings. Les deux premiers niveaux : premium et medium ont droit à un slot partenaire chacun parmi les conférences.

Au fait, pour info nous sommes montés à 59 conférences et non 49 en revoyant complètement l'agenda des jeudi et vendredi.

Devoxx sera l'occasion de rencontrer quelques rock stars mais aussi des jeunes talents. À quoi peut-on s'attendre, maintenant qu'on en a la liste ?

Difficile à dire, comme tu le dis il y aura des rocks stars et des jeunes talents. Ce que je trouve sympa c'est qu'il y aura surtout des gens d'horizons très divers, des jeunes issus de startups, des vieux briscards, des universitaires. Cette diversité devrait être très intéressante.

Comment seront préparés les orateurs ? Ce sera du free style ou, au contraire, tout sera contrôlé comme à TED, avec des maquilleurs, des costumiers et des répétitions ?

Non pas de costumiers ou de maquilleurs ;-) Après je ne suis pas devin et je ne peux pas prédire s'il y a des orateurs qui ont prévu des choses particulières.

Pourra-t-on parler aux orateurs entre les sessions ou seront-ils inaccessibles ?

Il n'y aura pas de couloirs privés ou de sas caché si c'est bien la question ;-) Les orateurs se baladeront donc dans les couloirs et je pense même que certains vont assister aux conférences. Après il est possible que certains choisissent de ne pas s'attarder, chacun sera libre de se déplacer comme il l'entend. Dans l'ensemble ils seront très accessibles.

Toutes les présentations seront intéressantes. Elles ont été choisies pour ça. Toutefois, peux-tu nous proposer une sélection personnelle des sessions à voir ?

Là c'est super compliqué comme question, j'ai mes propres sujets favoris qui ne vont pas forcément plaire à tous. D'ailleurs j'ai moi-même du mal à choisir car certains créneaux horaires proposent plusieurs talks intéressants en même temps.

Bref, je pose mon joker car je crois que c'est vraiment une affaire de goût personnel. Je rappelle que de toute façon chaque place donne droit à un abonnement Parleys afin de pouvoir revoir chaque conférence puisqu'elles seront toutes filmées.

On parle beaucoup de la nuit du jeudi. Que va-t-il se passer ? Vin à volonté ou quoi ?

Il s'agit de la soirée Meet and Greet. La soirée sera porte ouverte. Trois Sponsors ont la lourde responsabilité d'animer la soirée et je leur fais confiance pour qu'elle soit très sympa.

En parallèle il y aura des BOF qui permettront à quelques user groups de profiter des salles du Marriot. Il y a de grandes chances que l'ambiance soit très bonne !

5. Conclusions

Pourquoi venir à Devoxx France 2012 ?

Lorsque l'on n'est jamais venu dans une conférence, on ne se rend pas toujours compte de l'apport que cela peut amener mais imaginez toute la veille techno que vous réalisez toute l'année en lisant des blogs et des articles, multipliez ça par mille et vous aurez un vague aperçu de ce que Devoxx France représente.

Devoxx France c'est aussi la chance de croiser les types qui réalisent vos frameworks préférés et l'opportunité d'en manipuler si vous venez aux ateliers. Et si vous sentez un peu seul en tant que « développeur qui n'en veut », venez rencontrer vos semblables.

Comment convaincre mon boss de m'y envoyer ?

Devoxx France c'est une méga formation, c'est l'opportunité d'en revenir avec des tas d'idées neuves ou d'améliorations pour votre quotidien sur vos projets.

Et si votre boss vous parle de budget, sachez que c'est trois fois moins cher qu'une formation de trois jours et que c'est éligible au DIF.

Et si jamais cet argument a du poids, ce sera bon pour l'image de marque de savoir que votre boite envoie ses développeurs à Devoxx...

Retrouvez l'article de Thierry Leriche-Dessirier en ligne : Lien 47

Développement avancé avec Eclipse Zest

Cet article présente différents compléments sur la boîte à outils de visualisation de graphes sous Eclipse Zest. Il complète l'article d'introduction précédemment publié Introduction à Zest. On se propose ici d'enrichir le graphe que nous avions créé avec divers effets de mise en forme proposés par l'API de Zest.

1. Introduction

Dans l'article précédent, Introduction à Zest (Lien 48), nous avons vu comment construire un graphe, que ce soit à partir d'un modèle en utilisant les mêmes mécanismes que JFace, ou directement en créant nœuds et branches. Cet article reprend l'exemple de l'article précédent, et se propose de montrer comment enrichir le graphe de diverses manières : soit par le biais de styles personnalisés, de menus contextuels, ou encore par la mise en place d'un zoom, ou de layouts différents.

Ce tutoriel suppose que vous disposiez des connaissances de base sur les technologies suivantes :

- développement de plugins avec Eclipse : Lien 49 ;
- utilisation de Jface : Lien 50.

Par ailleurs, nous repartons sur le graphe orienté « branches » de l'article d'introduction, contenu dans la vue « LinkView ». Les sources sont disponibles ici (Lien 51) ou ici (Lien 52).

2. Changer l'algorithme d'agencement

La boîte à outils Zest possède plusieurs algorithmes d'agencement pour un graphe, qui modifient la disposition initiale du graphe lors de sa construction. Ces algorithmes sont :

- GridLayoutAlgorithm : dispose les éléments sur une grille, comme le ferait un GridLayout Swing classique ;
- HorizontalLayoutAlgorithm : dispose les éléments horizontalement, sur une seule ligne ;
- HorizontalShift : décale les éléments qui sont hors de la zone d'affichage du graphe vers la droite ;
- TreeLayoutAlgorithm : dispose les éléments selon un arbre vertical ;
- HorizontalTreeLayoutAlgorithm : effectue la même chose que le Tree LayoutAlgorithm, mais horizontalement ;
- RadialLayoutAlgorithm : place le nœud « racine » au centre et dispose les autres nœuds tout autour ;
- SpringLayoutAlgorithm : combine les éléments de manière à ce que les connexions aient approximativement la même taille et que le minimum de nœuds « dépassent » du graphe ;
- VerticalLayoutAlgorithm : dispose les éléments sur une droite verticale ;
- CompositeLayoutAlgorithm : combine différents algorithmes choisis à la construction pour le rendu du graphe. Par exemple, on peut utiliser le SpringLayout, puis HorizontalShift pour décaler les éléments qui seraient encore hors du graphe.

La plupart de ces layouts peuvent s'utiliser sans paramètres dans le constructeur, néanmoins, on peut utiliser l'option « LayoutStyles.NO_LAYOUT_NODE_RESIZING » afin que le layout ne redimensionne pas les nœuds du graphe. De même, l'option « LayoutStyles.ENFORCE_BOUNDS » force à contenir le graphe dans la zone d'affichage, sans dépassement.

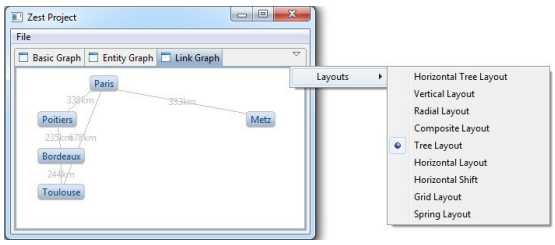
Nous allons mettre en œuvre ces différents algorithmes : pour cela complétez votre vue « LinkGraph » en ajoutant un menu localisé, qui permettra de choisir le layout à utiliser. Complétez le contenu de la méthode « createPartControl() » en y ajoutant le code suivant :

LinkGraphView.java

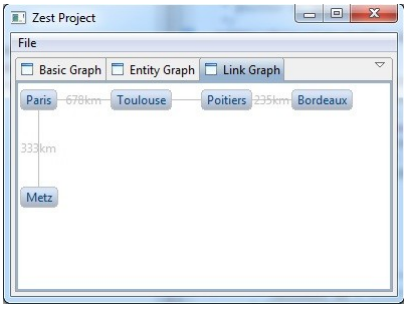
```
@Override
public void createPartControl(Composite parent) {

    //(...)

    //Creation de la liste des layouts
disponibles
    layouts = new HashMap<String,
LayoutAlgorithm>();
    layouts.put("Grid Layout", new
      GridLayoutAlgorithm(LayoutStyles.NO_LAYOUT_
NODE_RESIZING));
    layouts.put(« Horizontal Shift », new
      HorizontalShift(LayoutStyles.NO_LAYOUT_NODE
_RESIZING));
    layouts.put("Horizontal Layout", new
      HorizontalLayoutAlgorithm(LayoutStyles.NO_L
AYOUT_NODE_RESIZING));
    layouts.put("Horizontal Tree Layout", new
      HorizontalTreeLayoutAlgorithm(LayoutStyles.
NO_LAYOUT_NODE_RESIZING));
    layouts.put("Radial Layout", new
      RadialLayoutAlgorithm(LayoutStyles.NO_LAYOU
T_NODE_RESIZING));
    layouts.put("Spring Layout", new
      SpringLayoutAlgorithm(LayoutStyles.NO_LAYOU
T_NODE_RESIZING));
    layouts.put("Tree Layout", new
      TreeLayoutAlgorithm(LayoutStyles.NO_LAYOUT_
NODE_RESIZING));
    layouts.put("Vertical Layout", new
      SpringLayoutAlgorithm(LayoutStyles.NO_LAYOU
T_NODE_RESIZING));
    layouts.put("Composite Layout", new
CompositeLayoutAlgorithm( new
```

```
      LayoutAlgorithm[] {new
SpringLayoutAlgorithm(),
        new
HorizontalShift(LayoutStyles.NO_LAYOUT_NODE_RESIZ
ING)}));

    //Creation du menu de selection des layouts
    IMenuManager menuManager =
this.getViewSite().getActionBars()
        .getMenuManager();
    MenuManager layoutsMenu = new
MenuManager("Layouts");
    menuManager.add(layoutsMenu); // le menu de
selection apparait dans le menu
      localise

    Action a;
    for (final String s : layouts.keySet()) {
    //pour chaque layout on cree un element de
menu permettant de le
      selectionner
    a = new Action(s, IAction.AS_RADIO_BUTTON) {
        @Override
        public void run() {
        viewer.setLayoutAlgorithm(layouts.get(s),
true);
        }
    };
    layoutsMenu.add(a); //l'item est ajoute au
menu des layouts
    //par defaut, on selectionne le Spring Layout
    if (s.equals(« Spring Layout »)) {
        a.run();
        a.setChecked(true);
    }
    }

    // (...)

}
```

La vue dispose maintenant d'un menu localisé qui permet de sélectionner le layout à utiliser. Bien évidemment, suivant le type de données représentées par le graphe, il sera plus judicieux d'utiliser tel ou tel layout.



Tree Layout



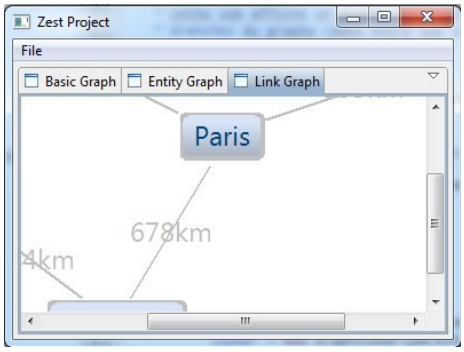
GridLayout

3. Mettre en place un zoom

Zest donne la possibilité de mettre en place très facilement un système de zoom sur le graphe. Pour cela, il faut que la vue (ou l'éditeur) implémente l'interface IZoomableWorkbenchPart. D'autre part, il faut aussi définir un menu, qui permettra à l'utilisateur de choisir son niveau de zoom. Reprenez le code de la vue « LinkGraph » et implémentez l'interface IZoomableWorkbenchPart. La méthode "getZoomableViewer()" doit retourner l'objet GraphViewer. D'autre part, ajoutez un menu dans le menu localisé de la vue, comme suit :

LinkGraphView.java

```
public class LinkGraphView extends ViewPart
implements IZoomableWorkbenchPart {

    // (...)

    @Override
    public void createPartControl(Composite
parent) {

     // (...)

    //Creation d'un menu pour zoomer sur le
graphe
    //ce menu sera affiche dans le menu localise
de la vue
    IMenuManager menuManager =
this.getViewSite().getActionBars()
        .getMenuManager();

    MenuManager zoomMenu = new
MenuManager("Zoom");
    ZoomContributionViewItem zoomItem = new
ZoomContributionViewItem(this);
    zoomMenu.add(zoomItem);

    menuManager.add(zoomMenu);

    // (...)

    }

    @Override
    public AbstractZoomableViewer
getZoomableViewer() {
    return viewer;
    }
}
```



Zoom

4. Interagir avec l'utilisateur

Au-delà de la visualisation, il peut être intéressant de

proposer à l'utilisateur une interaction avec le graphe, par exemple pour modifier le modèle. Dans notre cas, nous aimerions que l'utilisateur puisse ouvrir et fermer des routes. L'utilisateur doit pouvoir choisir le statut qu'il veut affecter à la route à partir d'un menu contextuel accessible depuis les branches du graphe. La première étape consiste à modifier le modèle pour rajouter dans la classe Route un attribut booléen "ouverte" et les getter/setter associés. Il faut ensuite créer le menu contextuel, au sein de notre classe LinkGraphView.java. Rajoutons le code suivant dans la méthode "createPartControl" :

LinkGraphView.java

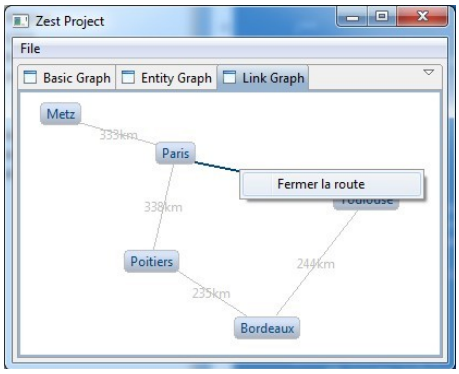
```
@Override
public void createPartControl(Composite parent) {

    // (...)

    //Creation du menu contextuel
    final MenuManager mgr = new MenuManager();
    viewer.getControl().setMenu(mgr.createContext
Menu(viewer.getControl()));
    mgr.setRemoveAllWhenShown(true); //le menu
est recree a chaque affichage
    mgr.addMenuListener(new IMenuListener() {
    @Override
    public void menuAboutToShow(IMenuManager
manager) {
        //le contenu du menu depend de la
selection courante
        IStructuredSelection selection =
(IStructuredSelection)
          viewer.getSelection();
        if (!selection.isEmpty() &&
selection.getFirstElement() != null) {
        Object first =
selection.getFirstElement();
        if (first instanceof Route) {
            manager.add(new
OpenRoadAction((Route)first));
        }
        }
    }
    });

    // (...)
}
```

Nous créons dans le menu une instance de "OpenRoadAction", qui permet de définir le comportement à adopter lorsque l'utilisateur la sélectionne. Cette classe est définie comme suit :

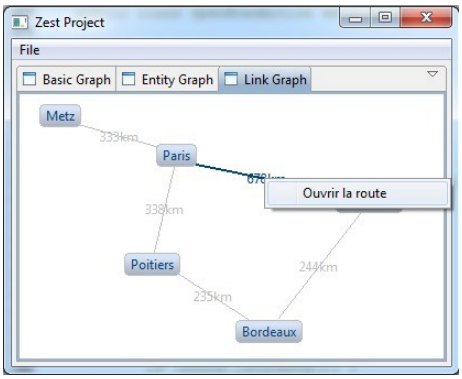
OpenRoadAction.java

```
/**
 * Cette action permet a l'utilisateur
d'ouvrir/fermer les routes
 * directement depuis le graphe.
 * @author A. Bernard
 */
public class OpenRoadAction extends Action {

    /**
     * la route geree par l'action
     */
    private Route route;

    /**
     * Constructeur.
     * Definit le texte de l'action affiche.
     * @param r la route selectionnee
     */
    public OpenRoadAction(Route r) {
    super();
    this.route = r;
    }

    @Override
    public String getText() {
    if (route.isOuverte()) {
        return "Fermer la route";
    } else {
        return "Ouvrir la route";
    }
    }

    @Override
    public void run() {
    if (route.isOuverte()) {
        route.setOuverte(false);
    } else {
        route.setOuverte(true);
    }
    //ne pas oublier de rafraichir le graphe
    viewer.refresh();
    }
}
```

On peut alors observer le résultat lors d'un clic droit sur le graphe :



Menu sur route ouverte



Menu sur route fermée

Notre graphe est maintenant interactif, mais rien ne permet de voir directement si une route est ouverte ou fermée. Nous allons donc dans la partie suivante étudier des interfaces fournies par Zest qui vont nous permettre d'afficher ces informations, en mettant en forme notre graphe.

5. Décorer son graphe

Afin de mettre en forme le graphe, il faut utiliser différentes interfaces, selon l'élément à modifier :

- l'interface "IConnectionStyleProvider" permet de modifier l'apparence des connexions ;
- l'interface "IEntityStyleProvider" permet de modifier l'apparence des entités ;
- l'interface "IEntityConnectionStyleProvider" permet de modifier l'apparence des connexions dans un graphe construit à partir de ses nœuds. En effet, rappelons-nous que dans ce cas, nous n'avons pas accès directement aux connexions.

Nous détaillons l'utilisation de ces interfaces dans les paragraphes suivants.

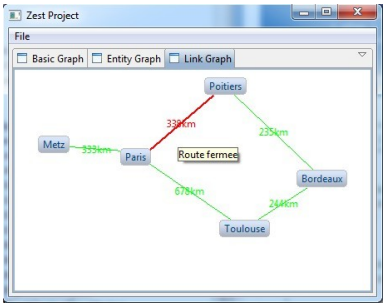
5.1. Mettre en forme les connexions

Dans un premier temps, nous allons retravailler sur la vue LinkGraphView : on se propose d'afficher en rouge les routes fermées et en vert les routes ouvertes. Reprenez le code de la classe LinkLabelProvider.java, et implémentez l'interface IConnectionStyleProvider. Complétez les nouvelles méthodes comme suit :

LinkLabelProvider.java

```
public class LinkLabelProvider extends
LabelProvider  implements
  IConnectionStyleProvider {

    // (...)

    /*
************************************************
*
     *    Methodes de l'interface
IConnectionStyleProvider
     *
************************************************
* */
    @Override
    public int getConnectionStyle(Object rel) {
    return ZestStyles.CONNECTIONS_SOLID; //valeur
par defaut
    //Utiliser ZestStyles.CONNECTION_DIRECTED
pour afficher une fleche
    // source -> destination
    }

    @Override
    public Color getColor(Object rel) {
    //Donne la couleur de base de la connexion
    if (rel instanceof Route) {
        Route r = (Route) rel;
        if (r.isOuverte()) {
        return
Display.getDefault().getSystemColor(SWT.COLOR_GRE
EN);
        } else {
        return
Display.getDefault().getSystemColor(SWT.COLOR_RED
);
        }
    } else {
        return null; //valeur par defaut
    }
    }

    @Override
    public Color getHighlightColor(Object rel) {
    //Donne la couleur de la connexion lorsque
cette derniere est
      selectionnee
    if (rel instanceof Route) {
        Route r = (Route) rel;
        if (r.isOuverte()) {
        return
Display.getDefault().getSystemColor(SWT.COLOR_GRE
EN);
        } else {
        return
Display.getDefault().getSystemColor(SWT.COLOR_RED
);
        }
    } else {
        return null; //valeur par defaut
    }
    }

    @Override
    public int getLineWidth(Object rel) {
    //Donne l'epaisseur d'une connexion
    return -1; //valeur par defaut
    }

    @Override
    public IFigure getTooltip(Object entity) {
    //Affiche un tooltip sur la connexion
    if (entity instanceof Route) {
        if (((Route)entity).isOuverte()) {
        return new Label("Route ouverte");
        } else {
        return new Label("Route fermee");
        }
    } else {
        return null;
    }
    }
}
```

Chaque méthode donne directement accès à la connexion en cours de construction, ce qui permet très facilement de définir le style des éléments. Nous pouvons observer le résultat sur notre graphe :



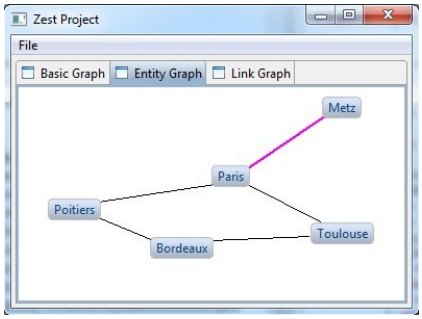
Mise en forme des connexions

Il est impossible d'utiliser cette méthode lorsque le graphe est construit à partir des nœuds. En effet, nous n'avons pas accès aux connexions lors de la construction du graphe. Pour cela, Zest propose l'interface IEntityConnectionStyleProvider. Reprenons la classe EntityLabelProvider :

EntityLabelProvider.java

```
public class EntityLabelProvider extends
LabelProvider implements
  IEntityConnectionStyleProvider {

    // (...)

    /*
**********************************************
*******
     *    Methodes de l'interface
IEntityConnectionStyleProvider
     *
**********************************************
******* */

    @Override
    public int getConnectionStyle(Object src,
Object dest) {
    return ZestStyles.CONNECTIONS_SOLID; //valeur
par defaut
    //Utiliser ZestStyles.CONNECTION_DIRECTED
pour afficher une fleche
    // source -> destination
    }

    @Override
    public Color getColor(Object src, Object
dest) {
    //Donne la couleur de la connexion
    return
Display.getDefault().getSystemColor(SWT.COLOR_BLA
CK);
    }

    @Override
    public Color getHighlightColor(Object src,
Object dest) {
    //Donne la couleur de la connexion si
selectionnee
    return
Display.getDefault().getSystemColor(SWT.COLOR_MAG
ENTA);
    }

    @Override
    public int getLineWidth(Object src, Object
dest) {
    //Donne l'epaisseur de la connexion
    return -1;
    }

    @Override
    public IFigure getTooltip(Object entity) {
    // Donne un tooltip pour le noeud selectionne
    return null;
    }
}
```

Chaque méthode permet d'accéder à la connexion définie par sa source et sa destination.



Mise en forme des connexions

5.2. Mettre en forme les nœuds

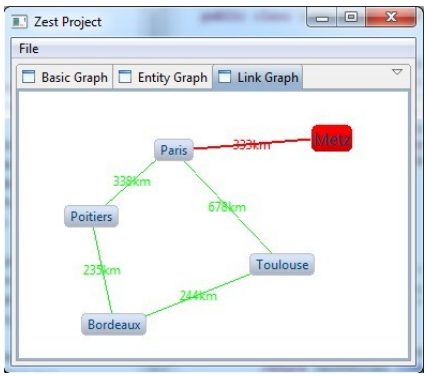
Nous souhaitons maintenant afficher en rouge les villes qui ne sont plus desservies par aucune route. Reprenez la classe LinkLabelProvider.java et implémentez l'interface IEntityStyleProvider :

LinkLabelProvider.java

```
>
public class LinkLabelProvider extends
LabelProvider  implements
  IConnectionStyleProvider, IEntityStyleProvider
{

    // (...)

    /*
*******************************************
     *    Methodes de l'interface
IEntityStyleProvider
     *
******************************************* */

    @Override
    public Color getNodeHighlightColor(Object
entity) {
    //Donne la couleur du noeud si selectionne
    return null;
    }

    @Override
    public Color getBorderColor(Object entity) {
    //Donne la couleur de la bordure des noeud
    return null; //valeur par defaut
    }

    @Override
    public Color getBorderHighlightColor(Object
entity) {
    //Donne la couleur de la bordure du noeud
selectionne
    return null; //Valeur par defaut
    }

    @Override
    public int getBorderWidth(Object entity) {
    // Donne l'epaisseur de la bordure du noeud
    return -1;
    }

    @Override
    public Color getBackgroundColour(Object
entity) {
    //Donne la couleur de fond par defaut des
noeuds
    if (entity instanceof Ville) {
        Ville v = (Ville) entity;
```

```
        //On regarde si les routes qui desservent
la ville sont toutes
        // fermees ou non
        boolean estReliee = false;
        for (Route r :
Model.INSTANCE.getRoutes()) {
        if (r.getSource().equals(v) ||
r.getDestination().equals(v)) {
            // La route consideree dessert la
ville.
            // Si elle est ouverte, la ville est
accessible par au
            // moins une route
            if (r.isOuverte()) {
            estReliee = true;
            }
        }
        }
        //Si la ville est reliee par au moins une
route, la couleur
        //affichee est celle par defaut, sinon le
noeud est colore en rouge
        if (estReliee) {
        return null;
        } else {
        return
Display.getDefault().getSystemColor(SWT.COLOR_RED
);
        }
    } else {
        return null;
    }
    }

    @Override
    public Color getForegroundColour(Object
entity) {
    //Donne la couleur d'avant-plan des noeuds
(le texte par ex.)
    return null; //valeur par defaut
    }

    @Override
    public boolean fisheyeNode(Object entity) {
    //Cree un zoom sur l'entite quand elle est
selectionnee
    return true;
    }
}
```

Une nouvelle fois, chaque méthode nous permet d'accéder très facilement à l'entité en cours d'affichage. Remarquez la méthode "fisheyeNode" qui permet d'effectuer un zoom sur le nœud lorsqu'il est survolé par la souris. Observons le résultat :



Mise en forme des entités

6. Filtrer certains éléments

Zest nous permet aussi de mettre en place des filtres sur les graphes, de la même manière que sur les viewers JFace. Créez le filtre suivant, qui nous permettra de cacher les villes qui ne sont desservies par aucune route :

EntityFilter.java

```
package com.abernard.zest.viewer;

import org.eclipse.jface.viewers.Viewer;
import org.eclipse.jface.viewers.ViewerFilter;

import com.abernard.zest.model.Model;
import com.abernard.zest.model.Route;
import com.abernard.zest.model.Ville;

/**
 * Ce filtre permet de cacher les villes
desservies par aucune route. Si
 * la ville n'est pas reliee par des routes, elle
n'est pas affichee.
 * @author A. Bernard
 */
public class EntityFilter extends ViewerFilter {

    @Override
    public boolean select(Viewer viewer, Object
parentElement, Object element)
      {
    // Determine si la ville doit etre affichee,
ou non
    if (element instanceof Ville) {
        Ville v = (Ville) element;
        //On regarde si les routes qui desservent
la ville sont toutes
        // fermees ou non
        boolean estReliee = false;
        for (Route r :
Model.INSTANCE.getRoutes()) {
        if (r.getSource().equals(v) ||
r.getDestination().equals(v)) {
            // La route consideree dessert la
ville.
            // Si elle est ouverte, la ville est
accessible par au
            // moins une route
            if (r.isOuverte()) {
            estReliee = true;
            }
        }
        }
        //Si la ville est reliee par au moins une
route, la ville est
        // affichee, sinon elle est filtree
        return estReliee;
    } else {
        return true;
    }
    }
}
```

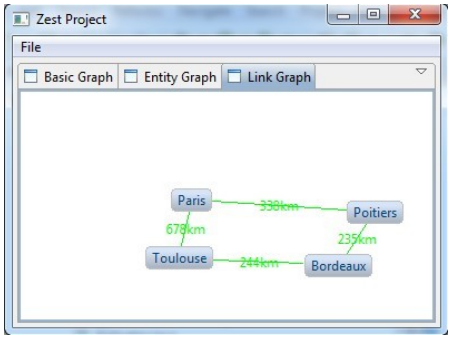
Ajoutez ce filtre au graphe dans la vue LinkGraphView.java :

LinkGraphView.java

```
@Override
public void createPartControl(Composite parent) {

    // (...)
```

```
    //Definition du filtre
    viewer.addFilter(new EntityFilter());

    // (...)
}
```

Observez maintenant le résultat lorsque l'on ferme la route qui relie Metz à Paris :



Filtre sur le graphe

7. Conclusion

Nous avons vu dans cet article comment modifier l'affichage du graphe et améliorer son ergonomie. Au terme de ces deux articles, nous pouvons constater que Zest est un outil de visualisation de graphes puissant tout en étant pratique et facile à utiliser, et qui propose assez d'options pour représenter des graphes clairs et précis, adaptés au modèle de chaque application.
Cette boîte à outils est notamment utilisée pour l'outil de visualisation de dépendances de PDE (Plugin Developpement Environment), actuellement dans l'Incubator d'Eclipse : PDE Incubator Dependency Visualization (Lien 53).

8. Liens utiles

Pour aller plus loin, voici quelques liens utiles :

- Tutoriels sur le développement de plugins Eclipse et l'utilisation de SWT/JFace : Lien 54 ;
- Page officielle de Zest : Lien 55 ;
- Wiki sur Zest : Lien 56 ;
- Snippets sur Zest : Lien 57.

Retrouvez l'article d'Alain Bernard en ligne : Lien 58.

Android : 850 000 activations par jour

L'Android Market triple son nombre d'applications en un an

Le Mobile World Congress (MWC) de Barcelone, l'événement majeur du monde du mobile a été l'occasion pour Google de faire le point sur Android.

Le système d'exploitation revendique une croissance de 250 % en un an avec actuellement 850 000 nouvelles activations de terminaux par jour contre 700 000 en fin décembre.

La plateforme enregistre une progression quasiment exponentielle et franchit le cap des 300 millions de dispositifs à travers le monde.

Selon Andy Rubin, en charge d'Android chez Google, ce chiffre n'inclut même pas les appareils comme le Kindle Fire d'Amazon qui n'utilisent pas les services de Google.

Succès incontestable qui fait de sa galerie d'applications l'une des plateformes les plus ciblées par les développeurs. L'Android Market voit son nombre d'applications triplé en seulement un an, passant de 150 000 à 450 000.

Stand Android au MWC

Seul domaine où le petit robot vert est encore à la traine : les tablettes. Android n'équipe actuellement que 12 millions de tablettes sur les 300 millions de dispositifs Android.

Commentez la news d'Hinault Romaric en ligne : Lien 57

L'intérêt des développeurs pour Android baisse en faveur de HTML 5

iOS demeure la plateforme préférée selon un rapport d'IDC et Appcelerator

Malgré le succès incontestable d'Anroid qui a franchi en fin du mois dernier le cap des 300 millions de dispositifs à travers le monde, la plateforme de Google attire peu les développeurs selon le dernier rapport du cabinet d'analyse IDC et Appcelerator.

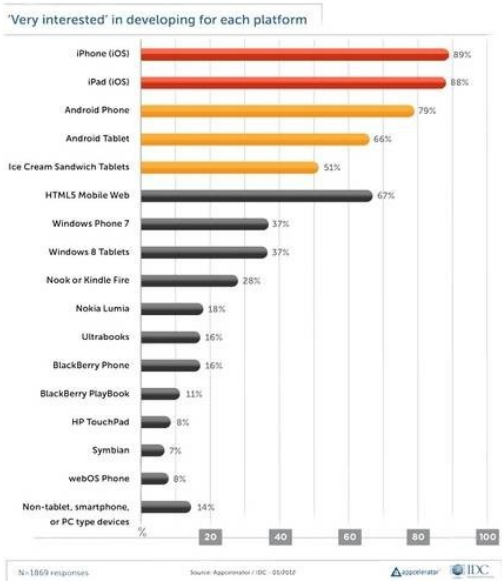
Le sondage mené auprès de 2173 développeurs du programme mobile Appcelerator sur leurs préférences et priorités de développement pour les jours à venir montre une baisse de l'intérêt de ceux-ci pour Android, passant de 85 % à l'été 2011 à 79 % en début 2012. Pour les tablettes Android, on constate également une baisse de 75 % à 66 % sur la même période.

Cette diminution de l'intérêt des développeurs pour Android serait due à la multiplication des versions de l'OS sur les différents terminaux qui finirait par lasser les développeurs.

« Nous estimons que beaucoup de développeurs sont mécontents de la fragmentation de la plateforme ainsi que du système de monétisation » a déclaré Mike King, responsable stratégie mobile chez Appcelerator. « Ces choses font qu'il est très difficile pour les développeurs de se faire de l'argent sur Android ».

iOS demeure la plateforme préférée des développeurs avec un taux d'intérêt de 89 % pour l'iPhone et 88 % pour l'iPad.

Cependant, avec l'arrivée du HTML5, les développeurs mobiles se désolidarisent de plus en plus des solutions natives pour les solutions Web. L'intérêt pour HTML 5 est estimé à 67 %.



L'orientation vers Windows Phone progresse lentement et l'OS mobile de Microsoft se trouve à 40 % devant RIM (BlackBerry Phone) qui enregistre une forte baisse à 16 %.

Commentez la news d'Hinault Romaric en ligne : Lien 59

« Ubuntu For Android » : un nouveau projet de Canonical

Pour porter son OS sur les smartphones

Ubuntu dans un téléphone ? Oui, c'est possible… si les constructeurs le décident.

Canonical, la société derrière la populaire distribution Linux, vient en effet d'annoncer un projet baptisé *« Ubuntu pour Android »*. Le principe en est simple mais pas évident.

En résumé, il sera possible d'installer un OS Ubuntu en s'appuyant sur le noyau d'Android. Le résultat sera un smartphone sous Android en mode téléphone, et un terminal sous Ubuntu en mode PC (sur le modèle du Atrix de Motorola).

Il ne s'agira donc pas d'une surcouche, d'un dual-boot ou d'une virtualisation mais d'un complément pour transformer un smartphone en véritable PC ayant accès à toute la logithèque des dépôts d'Ubuntu. Mais un Ubuntu qui s'appuiera toujours sur Android, notamment pour le carnet des contacts qui ne sera pas dupliqué, pour optimiser les fonctionnalités et la mémoire de l'appareil.

Cet *« Ubuntu pour Android »* sera présenté pour la première fois la semaine prochaine au Mobile World Congress de Barcelone. Quant à Canonical, la société affirme être déjà en pourparlers avec plusieurs constructeurs intéressés par cette solution.

Reste que tous les appareils ne pourront pas intégrer ce deuxième OS. Le minimum requis est en effet un téléphone disposant d'au minimum 2 Go de stockage rien que pour Ubuntu, d'un dual-core ARM 1 GHz et d'au minimum 500 Mo de RAM.

Un projet à ne pas confondre avec Boot2Gecko de la Fondation Mozilla (Lien 61), dont la démo sera également faite la semaine prochaine mais qui est lui un OS mobile à part entière.

Commentez la news de Gordon Fowler en ligne : Lien 62

Android Market est mort, vive Google Play

La nouvelle plateforme centralisera applications et services multimédias

Google play

Google vient d'ouvrir une nouvelle plate-forme : Google Play.

Son but est d'offrir aux utilisateurs un magasin en ligne d'applications, de musique, de livres et de films. Cette arrivée a pour effet la disparition de l'Androïd Market et d'autres services peu connus ici comme Google Music.

Ou pour le dire autrement, ces services seront intégrés et chapeautés par cette nouvelle appellation.

« À partir d'aujourd'hui, Android Market, Google Music et le eBookstore Google vont devenir une partie de Google Play » explique la société sur son blog officiel.

Présentation de Google Play : Lien 63

Au rayon des nouveautés donc, l'apparition des services musique, livres et films sur Google Play en Europe. Vous pouvez par exemple acheter un livre sur le site et le lire dans Google Chrome, ou bien sur un terminal Androïd.

Le service étant nouveau chez nous, les catalogues de livres, musique et films en français sont assez pauvres.

Pour ce qui est des applications, le Chrome Store est toujours actif pour Chrome et Chrome OS.

Si vous voulez télécharger une application Androïd, vous devrez vous connecter avec votre adresse mail Google utilisée avec votre tablette ou smartphone : Lien 64.

Copie d'écran de la page applications de Google Play

"Sur votre téléphone ou votre tablette Android, nous allons mettre à jour l'Android Market vers le Google Play Store au cours des prochains jours", promet Google sans donner plus de détails.

Google Play : Lien 65

Copie d'écran de la page musique de Google Play : Lien 66

Commentez la news de Yannick Inizan en ligne : Lien 67

Passage de paramètres en C#

Beaucoup de gens sont un peu désorientés par la façon dont les paramètres sont passés en C#, particulièrement en ce qui concerne les types référence. Cette page devrait aider à dissiper en partie cette confusion.
Microsoft a aussi une bonne page (Lien 68) à ce sujet (qui, je crois, utilise exactement la même terminologie que cette page-ci - faites-le moi savoir si elles semblent se contredire).
Note : Lee Richardson a écrit un article (Lien 69) complémentaire à celui-ci, spécialement pour ceux qui apprennent mieux avec des images. En gros il illustre les mêmes points, mais en utilisant de jolis diagrammes pour montrer ce qui se passe.

1. Traduction

Ceci est la traduction la plus fidèle possible de l'article de Jon Skeet, Parameter passing in C# : Lien 70.

2. Préambule : qu'est-ce qu'un type référence ?

En .NET (et donc en C#), il y a deux principales sortes de types : les *types référence* et les *types valeur*. Ils se comportent différemment, et une grande partie de la confusion sur le passage de paramètres est en réalité due au fait que certaines personnes ne comprennent pas la différence entre les deux. Voilà une rapide explication :

Un *type référence* est un type qui a pour valeur une référence aux données appropriées, plutôt que les données elles-mêmes. Par exemple, examinez le code suivant :

```
StringBuilder sb = new StringBuilder();
```

(J'ai utilisé StringBuilder comme un exemple quelconque de type référence - il n'a rien de particulier). Ici, on déclare une variable sb, on crée un nouvel objet StringBuilder, et on affecte à sb une référence à l'objet. La valeur de sb *n'est pas* l'objet lui-même, mais la référence. L'affectation des types référence est simple - la valeur qui est affectée est la valeur de l'expression ou variable - c'est-à-dire la référence. Ceci est démontré de façon plus poussée par cet exemple :

```
StringBuilder first = new StringBuilder();
StringBuilder second = first;
```

Ici on déclare une variable first, on crée un nouvel objet StringBuilder, et on affecte à first une référence vers l'objet. On affecte ensuite à la variable second la valeur de first. Cela signifie qu'elles font toutes les deux référence au même objet. Cependant, ce sont toujours elles-mêmes des variables indépendantes. Changer la valeur de first ne changera pas celle de second - cependant, tant que leurs valeurs sont des références vers le même objet, les changements faits sur l'objet via la variable first seront aussi visibles via la variable second. En voici un exemple :

```
StringBuilder first = new StringBuilder();
StringBuilder second = first;
first.Append("hello");
first = null;
Console.WriteLine(second);
```

Résultat :

```
hello
```

Ici on déclare une variable first, on crée un nouvel objet StringBuilder, et on affecte à first une référence vers l'objet. On affecte ensuite à la variable second la valeur de first. On appelle ensuite la méthode Append sur cet objet via la référence contenue dans la variable first. Après ça, on affecte null (une valeur qui ne fait référence à aucun objet) à la variable first. Enfin, on écrit le résultat de l'appel à la méthode ToString sur l'objet StringBuilder via la référence contenue dans la variable second. Cela affiche hello, ce qui montre bien que même si la valeur de first a changé, les données dans l'objet auquel elle faisait référence n'ont pas changé - et second fait toujours référence à cet objet.

Les types classe, les types interface, les types délégué et les types tableau sont tous des types référence.

3. Second préambule : qu'est-ce qu'un type valeur ?

Alors que les types référence ont un niveau d'indirection entre la variable et les données réelles, ce n'est pas le cas des *types valeur*. Les variables de type valeur contiennent directement les données. L'affectation d'un type valeur implique la copie des données réelles. Prenons une simple struct, par exemple :

```
public struct IntHolder
{
    public int i;
}
```

Partout où il y a une variable de type IntHolder, la variable contient toutes les données - en l'occurrence, une simple valeur d'entier. Une affectation copie la valeur, comme démontré ci-après :

```
IntHolder first = new IntHolder();
first.i = 5;
IntHolder second = first;
first.i = 6;
Console.WriteLine(second.i);
```

Résultat :
```
5
```

Ici, second.i a la valeur 5, parce que c'est la valeur de first.i quand l'affectation second=first est exécutée - les valeurs dans second sont indépendantes des valeurs dans first sauf au moment où l'affectation a lieu.

Les types simples (comme float, int, char), les types énumération et les types structure sont tous des types valeur.

Remarquez que de nombreux types (comme string) semblent à certains égards être des types valeur, mais sont en fait des types référence. Ils sont appelés *types immuables*. Cela signifie qu'une fois que l'instance a été construite, elle ne peut pas être modifiée. Cela permet à un type référence de se comporter de façon similaire à un type valeur à certains égards - en particulier, si vous avez une référence à un objet immuable, vous pouvez la renvoyer à partir d'une méthode ou la passer en paramètre d'une autre méthode, sans crainte que l'objet soit modifié dans votre dos. C'est pourquoi, par exemple, la méthode string.Replace ne modifie pas la chaîne sur laquelle elle est appelée, mais renvoie une nouvelle instance avec les nouvelles données de chaîne - si la chaîne originale était modifiée, toutes les autres variables contenant une référence à la chaîne verraient la modification, ce qui est rarement l'effet souhaité.

Remarquez la différence avec un type « *mutable* » (modifiable) comme ArrayList - si une méthode renvoie une référence d'ArrayList stockée dans une variable d'instance, le code appelant pourrait alors ajouter des éléments à la liste sans que l'instance ait son mot à dire, ce qui pose généralement un problème. Bien que les types référence immuables se comportent *comme* des types valeur, ce ne sont *pas* des types valeur, et il ne faut pas les considérer comme étant des types valeur.

Pour plus d'information sur les types valeur, les types référence, et où les données de chacun sont stockées en mémoire, veuillez vous référer à mon autre article à ce sujet : Lien 71.

4. Pour vérifier que vous avez compris le préambule...

À quel résultat vous attendriez-vous pour le code précédent si IntHolder était déclaré comme une classe plutôt que comme une structure ? Si vous ne comprenez pas pourquoi le résultat serait 6, veuillez relire les deux préambules, et envoyez-moi un e-mail si ce n'est toujours pas clair - si vous ne comprenez pas, c'est ma faute, pas la vôtre, et il faut que j'améliore cette page. Si vous comprenez, le passage de paramètres devient très facile à comprendre - lisez la suite.

5. Les différentes sortes de paramètres

Il y a quatre sortes de paramètres différents en C# : les paramètres par valeur (type par défaut), les paramètres par référence (qui utilisent le modificateur ref), les paramètres de sortie (qui utilisent le modificateur out), et les tableaux de paramètres (qui utilisent le modificateur params). *Vous pouvez utiliser n'importe quelle sorte de paramètre aussi bien avec des types valeur qu'avec des types référence.* Quand vous entendez les mots « référence » ou « valeur » (ou que vous les utilisez vous-même), il faut que ce soit très clair dans votre esprit s'il s'agit du fait qu'un paramètre est passé par référence ou par valeur, ou du fait que le type du paramètre est un type référence ou un type valeur. Si vous séparez bien les deux notions, elles sont très simples.

6. Paramètres par valeur

Par défaut, les paramètres sont passés par valeur. Cela signifie qu'un nouvel emplacement de stockage est créé pour la variable dans la déclaration de la fonction membre, et qu'il prend initialement la valeur spécifiée lors de l'invocation de la fonction membre. Si vous modifiez cette valeur, cela n'affecte pas les variables utilisées dans l'invocation. Par exemple, si nous avons :

```
void Foo(StringBuilder x)
{
    x = null;
}

...

StringBuilder y = new StringBuilder();
y.Append("hello");
Foo(y);
Console.WriteLine(y == null) ;
```

Résultat :
```
False
```

La valeur de y n'est pas modifiée du fait de l'affectation de null à x. Rappelez-vous cependant que la valeur d'une variable de type référence est la référence - si deux variables de type référence pointent vers le même objet, alors les modifications apportées aux données de l'objet seront visibles via les deux variables. Par exemple :

```
void Foo(StringBuilder x)
{
    x.Append(" world");
}

...

StringBuilder y = new StringBuilder();
y.Append("hello");
Foo(y);
Console.WriteLine(y);
```

Résultat :
```
hello world
```

Après l'appel à Foo, l'objet StringBuilder auquel y fait référence contient "hello world", puisque dans Foo les données " world" ont été ajoutées à cet objet via la référence contenue dans x.

Examinons maintenant ce qui se passe quand des types valeur sont passés par valeur. Comme je l'ai dit précédemment, la valeur d'une variable de type valeur est la donnée elle-même. En utilisant la définition précédente de la structure IntHolder, écrivons un code similaire à celui ci-dessus :

```
void Foo(IntHolder x)
{
```

```
    x.i = 10;
}

...

IntHolder y = new IntHolder();
y.i = 5;
Foo(y);
Console.WriteLine(y.i);
```

Résultat :

```
5
```

Quand Foo est appelée, x est initialement une structure avec la valeur i = 5. Sa valeur de i est ensuite changée en 10. Foo ne sait rien de la variable y, et après que la méthode Foo est terminée, la valeur dans y sera exactement la même qu'avant (c'est-à-dire 5).

Comme précédemment, vérifiez que vous comprenez bien ce qui se passerait si IntHolder était une classe plutôt qu'une structure. Vous devriez comprendre pourquoi y.i vaudrait 10 après l'appel à Foo dans ce cas.

7. Paramètres par référence

Les paramètres par référence ne passent pas les valeurs des variables utilisées pour l'invocation de la fonction membre - ils utilisent la variable elle-même. Plutôt que de créer un nouvel emplacement de stockage pour la variable dans la déclaration de la fonction membre, c'est le même emplacement qui est utilisé, donc la valeur de la variable dans la fonction membre et la valeur du paramètre par référence seront toujours la même. Les paramètres par référence requièrent le modificateur ref aussi bien dans la déclaration que dans l'invocation - cela implique qu'il est toujours clair que vous passez quelque chose par référence. Regardons les exemples précédents, en changeant juste le paramètre pour qu'il soit par référence :

```
void Foo(ref StringBuilder x)
{
    x = null;
}

...

StringBuilder y = new StringBuilder();
y.Append("hello");
Foo(ref y);
Console.WriteLine(y == null);
```

Résultat :

```
True
```

Ici, parce qu'une référence à y est passée plutôt que sa valeur, les modifications de la valeur du paramètre x sont immédiatement répercutées dans y. Dans l'exemple ci-dessus, y devient null. Comparez cela avec le résultat du même code sans le modificateur ref.

Examinons maintenant le code avec la structure que nous avions précédemment, mais en utilisant des paramètres par référence :

```
void Foo(ref IntHolder x)
{
    x.i = 10;
}

...

IntHolder y = new IntHolder();
y.i = 5;
Foo(ref y);
Console.WriteLine(y.i);
```

Résultat :

```
10
```

Les deux variables partagent le même emplacement de stockage, donc les modifications dans x sont également visibles via y, donc y.i a la valeur 10 à la fin de ce code.

7.1. Parenthèse : quelle est la différence entre passer un objet valeur par référence et passer un objet référence par valeur ?

Vous avez peut-être remarqué que le dernier exemple, où on passait une structure par référence, avait le même effet dans ce code que de passer une classe par valeur. Cela ne veut cependant pas dire que c'est la même chose. Examinez le code suivant :

```
void Foo(??? IntHolder x)
{
    x = new IntHolder();
}

...

IntHolder y = new IntHolder();
y.i = 5;
Foo(??? y);
```

Dans le cas où IntHolder est une structure (donc un type valeur) et où le paramètre est passé par référence (c'est-à-dire qu'on remplace ??? par ref ci-dessus), y devient une nouvelle valeur IntHolder - c'est-à-dire que y.i vaut 0. Dans le cas où IntHolder est une classe (donc un type référence) et où le paramètre est passé par valeur (c'est-à-dire qu'on enlève le ??? ci-dessus), la valeur de y n'est pas modifiée - c'est une référence vers le même objet qu'avant l'appel à la fonction membre. **Cette différence est absolument cruciale à la compréhension du passage de paramètres en C#, et c'est pourquoi je pense qu'il est extrêmement déroutant de dire que les objets sont passés par référence par défaut, plutôt que l'affirmation correcte qui est que les références d'objet sont passées par valeur par défaut.**

8. Paramètres de sortie

Comme les paramètres par référence, les paramètres de sortie ne créent pas un nouvel emplacement de stockage, mais utilisent l'emplacement de stockage de la variable spécifiée dans l'invocation. Les paramètres de sortie requièrent le modificateur out aussi bien dans la déclaration que dans de l'invocation - cela implique qu'il est toujours clair que vous passez quelque chose comme paramètre de sortie.

Les paramètres de sortie sont très similaires aux paramètres par référence. Les seules différences sont les suivantes :

- la variable spécifiée dans l'invocation n'a pas besoin d'être affectée avant d'être passée en paramètre de la fonction membre. Si la fonction membre se termine normalement, elle est considérée comme initialisée après le retour de la fonction (vous pouvez donc ensuite la « lire ») ;
- le paramètre est considéré comme non-initialisé (autrement dit, vous devez lui affecter une valeur avant de pouvoir le « lire » dans la fonction membre) ;
- la fonction membre *doit* affecter une valeur au paramètre avant de se terminer normalement.

Voici un exemple de code qui illustre cela, avec un paramètre int (int est un type valeur, mais si vous avez correctement compris les paramètres par référence, vous devriez être capable de voir quel serait le comportement avec un type référence) :

```
void Foo(out int x)
{
    // On ne peut pas lire x ici - il est considéré comme non-initialisé

    // Affectation - cela doit être fait avant que la méthode puisse se terminer normalement
    x = 10;

    // La valeur de x peut maintenant être lue
    int a = x;
}

...

// Déclaration d'une variable mais sans lui affecter de valeur
int y;

// Passage de la variable en tant que paramètre de sortie, bien qu'elle ne soit pas initialisée
Foo(out y);

// Une valeur a été affectée, on peut donc l'afficher
Console.WriteLine(y);
```

Résultat :

```
10
```

9. Tableaux de paramètres

Les tableaux de paramètres permettent de passer un nombre variable d'arguments à une fonction membre. La définition du paramètre doit inclure le modificateur params, mais il n'y a pas de mot-clé à spécifier pour l'utilisation du paramètre. Un tableau de paramètres doit se trouver à la fin de la liste des paramètres, et doit être un tableau à une dimension. Lors de l'utilisation de la fonction membre, n'importe quel nombre de paramètres (zéro inclus) peut apparaître dans l'invocation, tant qu'ils sont tous compatibles avec le type du tableau de paramètres. Une autre option est de passer un seul tableau, et dans ce cas le paramètre se comporte comme un paramètre par valeur normal. Par exemple :

```
void ShowNumbers(params int[] numbers)
{
    foreach(int x in numbers)
    {
        Console.Write(x + " ");
    }
    Console.WriteLine();
}

...

int[] x = {1, 2, 3};
ShowNumbers(x);
ShowNumbers(4, 5);
```

Résultat :

```
1 2 3
4 5
```

Dans la première invocation, la valeur de la variable x est passée par valeur, car le type de x est déjà un tableau. Dans la deuxième invocation, un nouveau tableau d'entiers est créé et contient les deux valeurs spécifiées, et une référence à ce tableau est passée (par valeur).

10. Miniglossaire

Quelques définitions informelles et rappels des termes utilisés :

- **fonction membre** : une fonction membre est une méthode, une propriété, un événement, un indexeur, un opérateur défini par l'utilisateur, un constructeur d'instance, un constructeur statique, ou un destructeur ;
- **paramètre de sortie** : un paramètre similaire à un paramètre par référence, mais avec des règles d'assignation définitive différentes ;
- **paramètre par référence** (sémantique de passage par référence) : un paramètre qui partage l'emplacement de stockage de la variable utilisée dans l'invocation de la méthode membre. Puisqu'ils partagent le même emplacement de stockage, ils ont toujours la même valeur (donc changer la valeur du paramètre change aussi la valeur de la variable utilisée dans l'invocation) ;
- **type référence**: type pour lequel la valeur d'une variable ou expression de ce type est une référence vers un objet plutôt que l'objet lui-même ;
- **emplacement de stockage** : zone de mémoire qui contient la valeur d'une variable ;
- **paramètre par valeur** (sémantique par défaut de passage par valeur) : paramètre qui a son propre emplacement de stockage, et donc sa propre valeur. Sa valeur initiale est celle de l'expression utilisée dans l'invocation de la fonction membre ;
- **type valeur** : type pour lequel la valeur d'une variable ou expression de ce type est les données de l'objet lui-même ;
- **variable** : nom associé à un emplacement de stockage et à un type. (Habituellement une seule variable est associée à un emplacement de stockage ; les exceptions sont les paramètres par référence et de sortie).

Retrouvez l'article de Jon Skeet traduit par Thomas Levesque en ligne : Lien 72.

Pourquoi les propriétés sont importantes

Dans le chapitre 8 (NdT : du livre C# in Depth), je suppose sans trop me poser de questions que les lecteurs considèrent comme une bonne pratique l'utilisation de propriétés plutôt que de champs. Eric m'a suggéré de mentionner pourquoi c'est une bonne pratique. Je trouvais que ça ne rentrait pas vraiment dans le chapitre, mais c'est un sujet qui revient souvent dans les groupes de discussion et listes de diffusion, donc je me suis dit que je pourrais en faire un article.

1. Traduction

Ceci est la traduction la plus fidèle possible de l'article de Jon Skeet, Why Properties Matter : Lien 73.

2. Quels choix allons-nous examiner ?

Il vaut mieux être clair sur le fait que cet article n'aborde pas la question de savoir si quelque chose doit être une méthode ou une propriété. Ce n'est pas toujours une décision évidente, et une discussion sur les mérites de l'une ou de l'autre serait intéressante mais détournerait l'attention de l'objet de cet article.

Le choix dont je parle dans cet article concerne la façon dont vous exposez les données : comme une propriété ou comme un champ non privé. C'est aussi simple que ça. Mon opinion personnelle est que presque aucun champ ne devrait être non privé. Je ferai une *petite* exception pour les champs statiques en lecture seule comme String.Empty, mais c'est (à peu près) tout.

Alors, si vous avez un champ non privé, pourquoi devriez-vous le rendre privé et exposer plutôt une propriété ? J'ai réparti mes arguments en trois catégories : problèmes de compatibilité, avantages pratiques de l'utilisation des propriétés, et des considérations plus philosophiques. Bien que les aspects philosophiques *semblent* moins importants, ce sont en fait ceux qui me causent le plus de préoccupation.

3. Problèmes de compatibilité

Les gens disent souvent que ça ne pose pas de problème d'utiliser des champs pour des données simples, et de les remplacer par des propriétés quand cela devient nécessaire. Mais...

3.1. Vous perdez la compatibilité binaire

Ce problème n'en est pas vraiment un si vous travaillez sur un projet pour lequel tous les binaires sont toujours livrés ensemble, ou si le champ est actuellement internal et que vous n'exposez pas les membres internes à d'autres assemblies. Cependant, si vous développez une bibliothèque de classes qui pourrait avoir besoin d'être mise à jour sans recompiler le code qui l'utilise, vous devriez comprendre que changer un champ en propriété est un *breaking change*. Cela inclut aussi la sérialisation.

3.2. Vous perdez la compatibilité des sources

On peut utiliser une propriété partout où on peut utiliser un champ, non ? Faux. On peut utiliser un champ pour un paramètre ref, alors qu'on ne peut pas (du moins en C#) utiliser une propriété de cette manière. Il y a d'autres cas subtils où il y a une différence entre les champs et les propriétés. Il est vrai que certains (la plupart ?) de ces cas concernent des structures mutables - une autre décision de conception à éviter dans presque toutes les situations - mais changer un champ en propriété peut parfois changer le comportement du code *sans même générer un avertissement ou une erreur*. Voici un exemple qui illustre cela, qui implique une structure mutable :

```
using System;

struct MutableStruct
{
    public int Value { get; set; }

    public void SetValue(int newValue)
    {
        Value = newValue;
    }
}

class MutableStructHolder
{
    public MutableStruct Field;
    public MutableStruct Property { get; set; }
}

class Test
{
    static void Main(string[] args)
    {
        MutableStructHolder holder = new
MutableStructHolder();
        // Affecte la valeur de  holder.Field
        holder.Field.SetValue(10);
        // Récupère holder.Property comme une
copie et modifie la copie
        holder.Property.SetValue(10);

        Console.WriteLine(holder.Field.Value);
        Console.WriteLine(holder.Property.Value);
    }
}
```

Les résultats sont différents : 10 pour la première ligne, ce qui montre que l'appel à holder.Field.SetValue a bien modifié la valeur dans holder.Field ; 0 pour la deuxième ligne, ce qui montre que l'appel à holder.Property.SetValue n'a *pas* changé la valeur dans holder.Property.

Si les lignes qui modifient les valeurs étaient simplement holder.Field.Value = 10 et holder.Property.Value = 10, le changement du champ en propriété serait un *breaking change* - la seconde ne compilerait pas, parce que l'expression holder.Property n'est pas classifiée comme une variable. C'est mieux que de changer le comportement silencieusement, mais ça reste un changement fâcheux.

3.3. Vous perdez la compatibilité avec la réflexion

La plupart du temps, on n'utilise pas la réflexion pour accéder à des membres auxquels on peut accéder directement. Cependant, si quelqu'un le fait, ce code ne fonctionnera plus s'il s'attend à trouver un champ et que celui-ci a été transformé en propriété de façon inattendue.

4. Avantages pratiques des propriétés

Il y a des circonstances où vous *pourriez* utiliser des champs non privés, parce que pour une raison ou une autre vous n'êtes pas concerné par les problèmes de compatibilité ci-dessus. Cependant, il y a quand même des avantages à utiliser des propriétés même pour des situations triviales :

- les propriétés permettent une granularité d'accès plus fine. Vous voulez que la propriété soit publiquement accessible en lecture, mais que l'accès en écriture soit protégé ? Pas de problème (du moins à partir de C# 2) ;
- vous voulez arrêter le débogueur à chaque fois que la valeur change ? Il suffit de mettre un point d'arrêt dans l'accesseur set ;
- vous voulez journaliser tous les accès ? Il suffit de mettre le code de journalisation dans l'accesseur get ;
- les propriétés sont utilisées dans la liaison de données , pas les champs.

Aucun des points ci-dessus ne correspond à l'usage traditionnel des propriétés qui consiste à ajouter de la logique, mais tous sont difficiles/impossibles à réaliser avec des simples champs. Vous *pourriez* utiliser des propriétés au cas par cas selon les besoins, mais pourquoi ne pas être cohérent dès le départ ? La question se pose encore moins avec les propriétés auto-implémentées de C# 3.

5. La raison philosophique de n'exposer que des propriétés

Pour chaque type que vous écrivez, vous devriez examiner son interface avec le reste du monde (y compris les classes dans le même assembly). C'est la description de ce qu'il rend disponible, son apparence externe. L'*implémentation* ne devrait pas faire partie de cette description au-delà de ce qui est absolument nécessaire (c'est pourquoi je préfère la composition à l'héritage, quand le choix a un sens - l'héritage expose ou limite généralement les implémentations possibles).

Une propriété communique l'idée que « je mettrai une valeur à votre disposition, ou j'accepterai une valeur que vous me fournirez ». Ce n'est pas un concept d'implémentation, c'est un concept d'interface. Un champ, au contraire, communique l'implémentation - il dit « ce type représente une valeur de cette façon très spécifique ». Il n'y a pas d'encapsulation, c'est le format de stockage brut. C'est une des raisons pour lesquelles les champs ne font pas partie des interfaces - ils n'y ont pas leur place, car ils indiquent *comment* quelque chose est réalisé plutôt que *ce qui* est réalisé.

Je suis plutôt d'accord sur le fait que dans beaucoup de cas, des champs *pourraient* être utilisés sans problème pendant la durée de vie d'une application. Il est juste difficile de savoir à l'avance quels sont ces cas, et cela viole toujours le principe de conception qui consiste à ne pas exposer l'implémentation

6. Cas vraiment exceptionnels

Chaque règle a bien sûr des exceptions. Rico Mariani est un type très malin, qui s'y connaît très bien en matière de performance, et qui connaît certainement les principes de conception aussi. Dans un billet de blog à propos des primitives graphiques (Lien 74), il détaille pourquoi il a écrit une structure mutable de plus de 16 octets, qui expose tous ses champs publiquement - enfreignant trois importantes recommandations de conception dans le même type. *N'utilisez pas cet exemple comme une raison pour exposer des champs publiquement* - en tout cas, pas à moins que les mêmes justifications s'appliquent aussi. Les experts peuvent se permettre d'enfreindre les règles dans quelques rares cas, parce qu'ils savent exactement dans quoi ils s'engagent. Les simples mortels comme moi - et comme la plupart des lecteurs de cette page, probablement - devraient au moins passer une nuit blanche à réfléchir à leurs raisons avant d'enfreindre ces recommandations.

Il est intéressant de noter que l'une des raisons de Rico pour exposer des champs est en fait la même que ma raison précédente pour préférer des propriétés - cela expose l'implémentation. Occasionnellement, il est souhaitable de donner à l'appelant la garantie que l'implémentation restera toujours telle qu'elle est - et c'était le cas ici. Je dois dire que je n'ai jamais eu besoin de fournir ce genre de garanties dans le code que j'ai écrit, mais on peut trouver toutes sortes de situations.

Un autre cas exceptionnel que *j'ai* utilisé est de rendre des champs internal dans un type imbriqué privé. Puisque le type est privé, le fait qu'il s'agit d'un champ ne sera jamais exposé à d'autres types que celui dans lequel il est imbriqué - et qui « possède » effectivement le code du type imbriqué de toute façon. Avec C# 3, j'utiliserai à l'avenir des propriétés auto-implémentées de toute façon - avant c'était juste plus pratique d'utiliser des champs.

7. Conclusion

Dans presque tous les cas, les champs devraient être privés. Pas seulement non publics, mais privés. Avec les propriétés auto-implémentées en C# 3, ça ne coûte pratiquement plus rien en lisibilité ou en quantité de code d'utiliser une propriété plutôt qu'un champ. En de *rares* occasions, vous pourriez trouver une vraie bonne raison d'utiliser un champ, mais réfléchissez-y longuement et sérieusement avant de valider ce choix, surtout si vous allez l'exposer au-delà de l'accès internal.

Retrouvez l'article de Jon Skeet traduit par Thomas Levesque en ligne : Lien 75

Casse-têtes en C#

Régulièrement, je tombe sur des situations intéressantes en C# qui amènent des résultats surprenants. Cette page contient une liste d'exemples. Pour ceux ne contenant qu'un bout de code, nous supposerons que celui-ci est dans la méthode Main. Afin de ne pas tomber accidentellement sur les résultats avant que vous ne le souhaitiez, j'ai mis les réponses sur une autre page : Lien 76.

1. Traduction

Ceci est la traduction la plus fidèle possible de l'article de Jon Skeet, C# Brainteasers : Lien 77.

2. Surcharge

Qu'est-ce qui est affiché, et pourquoi ?

```
using System;

class Base
{
    public virtual void Foo(int x)
    {
        Console.WriteLine ("Base.Foo(int)");
    }
}

class Derived : Base
{
    public override void Foo(int x)
    {
        Console.WriteLine ("Derived.Foo(int)");
    }

    public void Foo(object o)
    {
        Console.WriteLine
("Derived.Foo(object)");
    }
}

class Test
{
    static void Main()
    {
        Derived d = new Derived();
        int i = 10;
        d.Foo(i);
    }
}
```

3. Ordre ! Ordre !

Qu'est-ce qui va s'afficher, pourquoi, et en êtes-vous sûr ?

```
using System;

class Foo
{
    static Foo()
    {
        Console.WriteLine ("Foo");
    }
}

class Bar
{
    static int i = Init();

    static int Init()
    {
        Console.WriteLine("Bar");
        return 0;
    }
}

class Test
{
    static void Main()
    {
        Foo f = new Foo();
        Bar b = new Bar();
    }
}
```

4. Bête arithmétique

Les ordinateurs sont censés être bons en calcul, n'est-ce pas ? Alors pourquoi la console renvoie-t-elle "False" ?

```
double d1 = 1.000001;
double d2 = 0.000001;
Console.WriteLine((d1-d2)==1.0);
```

5. Print, print, print...

Voici un code utilisant la fonctionnalité de méthode anonyme de C# 2. Que fait-il ?

```
using System;
using System.Collections.Generic;

class Test
{
    delegate void Printer();

    static void Main()
    {
        List<Printer> printers = new
List<Printer>();
        for (int i=0; i < 10; i++)
        {
            printers.Add(delegate
{ Console.WriteLine(i); });
        }

        foreach (Printer printer in printers)
        {
            printer();
        }
    }
}
```

6. Il n'y a littéralement aucun problème avec le compilateur ici...

Est-ce que ce code pourrait compiler ? Compile-t-il ?

Qu'est-ce que cela signifie ?

```
using System;

class Test
{
    enum Foo { Bar, Baz };

    static void Main()
    {
        Foo f = 0.0;
        Console.WriteLine(f);
    }
}
```

En voici d'autres dans le même genre...

```
using System;

class Test
{
    enum Foo { Bar, Baz };

    const int One = 1;
    const int Une = 1;

    static void Main()
    {
        Foo f = One-Une;
        Console.WriteLine(f);
    }
}
```

7. Inférence de type à gogo

J'ai d'abord vu ceci sur le blog d'Ayende (sous une forme un peu plus obscure, il est vrai) : Lien 78. Encore une fois, réfléchissez sur ce que le code va afficher et pourquoi.

```
using System;

class Test
{
    static void Main()
    {
        Foo("Hello");
    }

    static void Foo(object x)
    {
        Console.WriteLine("object");
    }

    static void Foo<T>(params T[] x)
    {
        Console.WriteLine("params T[]");
    }
}
```

Retrouvez l'article de Jon Skeet traduit par Jean-Michel Ormes en ligne : Lien 79

Web sémantique

Les dernières news

Google bientôt capable de répondre aux questions ?

La société implémentera la recherche sémantique dans son moteur de recherche

Google prépare une importante mise à jour de son moteur de recherche.

Selon les déclarations d'Amit Singhal, un haut cadre de la société Google, au Wall Street Journal, l'éditeur est sur le point d'apporter la mise à jour la plus importante de l'histoire de son moteur de recherche, qui pourrait affecter des millions de sites dont le classement dépend du PageRank. Cette mise à jour permettra au moteur de faire face à la concurrence et de s'accommoder des nouvelles technologies.

La mise à jour offrira au moteur de nouvelles aptitudes qui permettront à celui-ci de ne plus afficher uniquement des liens vers des pages traitant d'une question, mais de formuler directement une réponse exacte.

Les ingénieurs de Google se sont appuyés sur la recherche sémantique pour que le moteur puisse déterminer le sens exact de la requête de l'internaute. Une énorme base de données contenant des centaines de millions d'entités comme les personnes, les lieux ou des choses a été constituée à cet effet.

Ainsi, lorsqu'un utilisateur va lancer une recherche pour "Lake Tahoe" par exemple, il obtiendra directement des informations comme l'emplacement, la latitude, la température moyenne et la teneur en sel. Rapporte le Wall Street Journal.

D'après le quotidien, cette mise à jour vise également à contrer Facebook qui pourrait éventuellement proposer un moteur de recherche qui utilise sa base de données qui contient des millions d'informations sur des personnes, des lieux, etc.

Le passage à la recherche sémantique pourrait influencer de 10 à 20 % les résultats des recherches, selon les déclarations d'une personne proche du dossier au WSJ.

La mise à jour se fera progressivement dans les mois à venir et pourra s'achever en 2013. L'éditeur n'a donné aucun détail sur le calendrier de déploiement de celle-ci.

Commentez la news d'Hinault Romaric en ligne : Lien 80.

Les derniers tutoriels et articles

Nouveautés de SPARQL 1.1

La nouvelle version de SPARQL amènera un très grand nombre de nouveautés ; dont voici un aperçu.

1. Introduction

La nouvelle version de SPARQL amènera un très grand nombre de nouveautés. Nous étudierons certaines de ces nouveautés en les illustrant par des exemples simples.

2. Les projections

Il s'agit d'assignations et de créations de nouvelles valeurs aussi utilisables avec les agrégats, les fonctions mathématiques et la bibliothèque de fonctions.

Exemple :

le prix de tous les articles en ajoutant la TVA

```
SELECT (?prix * (19.6/100) AS ?prixTVA)
WHERE {
        ?article :prix ?prix
}
```

Il faut bien faire attention à ne pas utiliser la même variable des deux côtés de AS. Par exemple il est impossible d'écrire ?prix * (19.6/100) AS ?prix.

3. Les négations

Précédemment pour faire une négation en SPARQL, on utilisait FILTER et BOUND. Avec ces mots-clefs le nom de toutes les personnes qui n'ont pas de date d'anniversaire s'obtenait ainsi :

```
SELECT ?name
WHERE {
        ?x foaf:givenName ?name .
        OPTIONAL { ?x foaf:birthday ?date } .
        FILTER (!bound(?date))
}
```

Maintenant, avec SPARQL 1.1, on dispose de deux

nouveaux mots clefs : MINUS et NOT EXISTS.
NOT EXISTS a été défini pour deux raisons : simplifier la négation en SPARQL 1.1 et identifier les patrons de requête non existants. Voici un exemple simple d'utilisation :

```
SELECT ?name
WHERE {
        ?x foaf:givenName ?name .
        FILTER NOT EXISTS {
                ?x dc:date ?date
        }
}
```

MINUS a aussi été défini pour deux raisons : simplifier la négation en SPARQL 1.1 et supprimer certaines valeurs des résultats (évaluer le MINUS et le soustraire des résultats). On peut l'utiliser dans le même exemple :

```
SELECT ?name
WHERE {
        ?x foaf:givenName ?name .
        MINUS {
                ?x dc:date ?date
        }
}
```

Quelle est la différence entre les deux ? Tout d'abord, le nommage des variables. Avec le triplet suivant :

```
:a :b :c .
```

la requête :

```
SELECT * {
        ?s ?p ?o .
        FILTER NOT EXISTS {
                ?x ?y ?z
        }
}
```

évalue un jeu de résultats sans solutions car {?x ?y ?z} ne correspond à aucune donnée de ?s ?p ?o, donc NOT EXISTS {? x? y? z} élimine toutes les solutions. Il n'y a donc aucun résultat, tandis qu'avec MINUS, il n'y a pas de variable partagée entre la première partie (?s ?p ?o) et la seconde (?x ?y ?z). Aucune liaison n'est éliminée, et cela donne comme résultat a b c. La requête s'écrit :

```
SELECT * {
        ?s ?p ?o .
        MINUS {
                ?x ?y ?z
        }
}
```

4. L'agrégation

Avec SPARQL 1.0 on devait utiliser un script externe pour les fonctions agrégats « classiques » et disponibles en SQL ; comme compter un nombre de résultats, trouver une valeur ou moyenne, ou même prendre une valeur aléatoire. Maintenant, avec SPARQL 1.1, quelques agrégats sont fournis par défaut. Ces agrégats sont COUNT, SUM, MIN, MAX, AVG, SAMPLE, GROUP_CONCAT, ainsi qu'une fonctionnalité de groupage GROUP BY et une fonction de filtrage des groupes HAVING. Tous ces agrégats sont compatibles avec les projections. Dans le détail :

- COUNT : nombre d'éléments associés à une expression ;
- SUM : somme des valeurs associées à une expression ;
- MIN : élément ayant la plus petite valeur dans l'ensemble des éléments associés à une expression ;
- MAX : élément ayant la plus grande valeur dans l'ensemble des éléments associés à une expression ;
- AVG : moyenne de l'ensemble des éléments associés à une expression ;
- SAMPLE : valeur aléatoire dans un jeu de données ;
- GROUP_CONCAT : concaténer les résultats d'une agrégation ;
- GROUP BY : afin de calculer les valeurs agrégées pour une solution, celle-ci est d'abord divisée en un ou plusieurs groupes, et la valeur globale est calculée pour chaque groupe ;
- HAVING : agit sur des ensembles de solutions groupées, de la même manière que FILTER agit sur des solutions non groupées.

Exemple avec SUM, GROUP BY et HAVING :

Le total de tous les éléments et le prix de chaque groupe d'éléments ne doit pas dépasser 10 euros

```
SELECT (SUM(?prix AS ?prix_total))
WHERE {
        ?element :prix ?prix .
}
GROUP BY ?element
HAVING (SUM(?prix) &gt; 10)
```

Les autres agrégats *COUNT*, *MIN*, *MAX*, *AVG*, *SAMPLE* et *GROUP_CONCAT* s'utilisent de la même façon.

5. Les chemins

Un chemin est une voie possible entre deux nœuds dans un graphe. Ce système est basé sur les expressions régulières. Celui-ci peut par exemple être utilisé pour identifier les amis d'un ami et les amis de leurs amis, aussi loin qu'on le veut et quelle que soit la distance. Ou encore pour identifier les sous-catégories d'un terme dans une source de données, et ceci peu importe la profondeur à laquelle il se trouve. Ou même encore pour identifier si deux ressources sont liées par des liens, tout en excluant ceux qui ne sont pas intéressants. Voici les éléments que ce système propose :

- iri : une IRI ou un nom préfixé, un chemin de longueur unitaire ;
- ^elt : chemin inverse (de l'objet au sujet) ;
- !iri ou !(iri1| ...|irin) : négation d'un ensemble de propriétés. Une IRI qui n'est pas une des irin. !iri est un raccourci pour !(iri) ;
- !^iri ou !(iri1| ...|irij|^irij+1| ...|^irin)) : négation d'un ensemble de propriétés avec certaines propriétés inverses. Une IRI qui n'est pas une des irin, ni l'une des irij+1...irin des chemins inverses. !^iri est un raccourci pour !(^iri) ;

- (elt) : un groupe de chemin elt, des crochets de contrôle prioritaire ;
- elt1 / elt2 : une séquence de chemin de elt1, suivie par elt2 ;
- elt1 | elt2 : une voie alternative de elt1 ou elt2 (toutes les possibilités sont essayées) ;
- elt* : un chemin de zéro ou plusieurs occurrences de elt ;
- elt+ : un chemin d'une ou plusieurs occurrences de elt ;
- elt? : un chemin de zéro ou un elt ;
- elt{n,m} : un chemin entre n et m occurrences de elt ;
- elt{n} : exactement n occurrences de elt ;
- elt{n,} : n occurrences ou plus de elt ;
- elt{,n} : entre 0 et n occurrences de elt.

Comment peut-on utiliser ces syntaxes ?

Une ou deux possibilités

```
{ :book1 dc:title|rdfs:label ?displayString }
```

Trouver le nom de toutes les personnes que connaît Alice.

```
{
        ?x foaf:mbox <mailto:alice@example> .
        ?x foaf:knows/foaf:name ?name .
}
```

Trouver le nom des personnes qui connaissent les personnes que connaît Alice

```
{
        ?x foaf:mbox <mailto:alice@example> .
        ?x foaf:knows/foaf:knows/foaf:name ?
name .
}
```

D'une autre façon :

```
{
        ?x foaf:mbox <mailto:alice@example> .
        ?x foaf:knows{2}/foaf:name ?name .
}
```

Ces deux dernières requêtes sont les mêmes : la seconde est simplement l'inversion du sens de la propriété qui échange les rôles de sujet et objet.

```
{ ?x foaf:mbox <mailto:alice@example> }
{ <mailto:alice@example> ^foaf:mbox ?x }
```

Trouver toutes les personnes qui connaissent quelqu'un qui connaît ?X

```
{
        ?x foaf:knows/^foaf:knows ?y .
        FILTER(?x != ?y)
}
```

Trouver les noms de toutes les personnes qui peuvent être atteintes en partant de Alice par le biais de foaf:knows

```
{
      ?x foaf:mbox &lt;mailto:alice@example&gt; .
      ?x foaf:knows+/foaf:name ?name .
}
```

Certaines formes d'inférence limitée sont aussi possibles.

Trouver tous les types et supertypes d'une ressource

```
{ &lt;http://example/thing&gt;
rdf:type/rdfs:subClassOf* ?type }
```

6. Les requêtes imbriquées

Ce type de requête nécessitait de passer par un langage externe à SPARQL. Il est maintenant possible d'imbriquer les requêtes les unes dans les autres. Néanmoins, il faut faire très attention à la portée des variables. Petit exemple :

Tous les articles et le nom des auteurs triés par ordre alphabétique

```
SELECT ?article ?name
WHERE {
        ?article :author ?author .
        {
                SELECT ?name WHERE {
                        ?author foaf:name ?name
                } ORDER BY ?name
        }
}
```

Le modificateur ORDER BY trie les éléments donnés en paramètre suivant l'opérateur « < ». On peut lui dire si l'on veut un ordre croissant ou décroissant avec les fonctions ASC() et DESC().

7. Une bibliothèque de fonctions

Ces fonctions étaient souvent implémentées par certains points d'accès SPARQL mais pas standardisées, ce qui provoquait la création de requêtes non standard. Ainsi une longue liste d'opérateurs et de fonctions on été ajoutés dans SPARQL 1.1. On peut en trouver la liste complète dans le standard.

8. Mise à jour du langage et du protocole

L'ajout le plus important est la possibilité de modifier les graphes. Jusqu'à présent, les données étaient seulement accessibles en lecture et ne pouvaient pas être mises à jour ; cela nécessitait de passer par des langages externes. Deux types d'opérations ont été définis : les opérations sur les graphes (graph management) et les opérations sur les données au sein de graphes (graph update). Les opérations possibles sur les graphes sont :

- créer et supprimer des graphes : CREATE et DROP ;
- ajout/suppression de triplets : LOAD et CLEAR.

Pour les données et les triplets :

- insertion de données sans variables : INSERT DATA ;
- suppression de données sans variables : DELETE DATA ;
- insertion de triplets dans un graphe : INSERT ;
- suppression de triplets dans un graphe : DELETE.

Il va de soi que le protocole a aussi été modifié en conséquence via HTTP RDF Update (RESTful).

9. Les descriptions de services

Ainsi un point d'accès SPARQL pourra fournir des

informations à son sujet. Avec une simple requête il sera donc possible de connaître :

- les fonctions disponibles ;
- la quantité de données ;
- les vocabulaires utilisés ;
- les informations techniques concernant ce point d'accès...

Tout ceci sera bien entendu représenté par un vocabulaire spécifique.

10. Inférence sur les données

SPARQL sera associé avec les inférences de langages pouvant définir des vocabulaires ou des règles tels que RDFS, OWL2 ou RIF.

Par exemple, on sait que le rang de la propriété foaf:knows est la classe foaf:Person. On admet que la personne :julien connaisse une personne :thibaut. Soit la requête suivante :

```
SELECT ?x
WHERE {
        ?x a foaf:Person .
}
```

Avec SPARQL 1.0, seul :julien ferait partie du résultat. Avec l'inférence, ici de RDFS, SPARQL 1.1 donnera en sortie :julien et :thibaut.

11. Les requêtes distribuées

Avant on utilisait FROM et FROM NAMED pour interroger des graphes distants. Avec SPARQL 1.1 il devient possible d'interroger des points d'accès SPARQL distants. Le mot clef SERVICE est utilisé pour interroger les points d'accès distants et il est possible de combiner ces services. Par exemple :

```
SELECT ?person
WHERE {
        SERVICE &lt;http://dbpedia.org/sparql&gt;
{?s a foaferson .}
        SERVICE &lt;http://example.org/sparql&gt;
{?s a foaferson .}
}
```

Le mot clef BINDINGS propage des contraintes sur différents points d'accès, ce qui simplifie le FILTER. Par exemple, si on veut toutes les personnes portant le nom « julien » dans deux jeux de données (points d'accès SPARQL) différents :

```
SELECT ?person WHERE {
        SERVICE <http://dbpedia.org/sparql> {
                ?p a foaf:Person ;
                foaf:givenName ?name
        }
        SERVICE <http://example.org/sparql> {
                ?p a foaf:Person ;
                foaf:givenName ?name
        }
}
BINDINGS ?name { ("julien") }
```

12. Référence

Si vous voulez avoir plus de détails sur ces nouveautés, consultez le standard : Lien 81.

Retrouvez l'article de Julien Plu en ligne : Lien 82

Nouveautés de RDF 1.1

La nouvelle version de RDF présentera certes un très petit nombre de nouveautés mais celles-ci sont très avantageuses et importantes.

1. Support pour les graphes multiples et magasins de graphes

La communauté RDF utilise le terme « graphes nommés » depuis plusieurs années dans divers milieux, mais ce terme reste ambigu et fait souvent référence à ce qui est plutôt des graphes cités, des graphes littéraux, des URI pour les graphes, des bases de connaissances, des magasins de graphes... Le terme « support pour les graphes multiples et magasins de graphes » est utilisé comme un terme neutre qui n'est pas et ne doit pas être considéré comme définitif. Le groupe de travail de RDF définira le(s) terme(s) exact(s) et s'occupera de standardiser cela.

2. Syntaxe

2.1. Fonctionnalités ajoutées

La plus importante est très certainement la normalisation de la syntaxe RDF Turtle. Cette syntaxe devra supporter les graphes multiples et les magasins de graphes.

La seconde est de définir et standardiser une syntaxe JSON pour RDF. Le but de cette standardisation est de fournir une sérialisation RDF aussi complète que possible incluant, comme pour Turtle, le support des graphes multiples et des magasins de graphes. Cependant, ces caractéristiques peuvent être ignorées et des caractéristiques de syntaxes spéciales peuvent être introduites si cela peut faciliter l'adoption de JSON par la communauté des développeurs d'applications Web.

2.2. Fonctionnalités prévues

Une des plus demandées est de standardiser des ajouts à RDF/XML pour répondre à certaines voire à toutes les fonctionnalités des syntaxes Turtle et JSON, qui ne sont actuellement pas exprimables directement en RDF/XML et qui, si on essaye de les exprimer dégradent les systèmes RDF/XML existants. Les candidats sont les graphes multiples et les magasins de graphe ou des listes.

Une autre idée de fonctionnalité est de standardiser un sous-ensemble de RDF/XML adapté au traitement, avec des outils standard XML comme XSLT et XQuery (par exemple, pas de nœuds typés, pas de nidification, etc.)

3. Changements/modification des concepts, du modèle ou de la sémantique de RDF

3.1. Fonctionnalités ajoutées

La plus attendue est la clarification de l'usage des IRI (International Resource Identifier) comme référence pour les ressources RDF. Une autre encore est de compléter les règles d'inférence qui sont actuellement incomplètes, et de mettre à jour le lemme d'implication.

Dernièrement une importante discussion eut lieu sur la mailing list de RDF afin de déprécier certaines fonctions RDF (par exemple : la réification, les conteneurs). « Déprécier » dans ce contexte signifie que les nouveaux déploiements, comme les éditeurs de données, sont invités à ne pas utiliser la(les) fonctionnalitée(s) concernée(s). Les systèmes conformes (analyseurs, etc.) seront encore nécessaires pour implémenter la fonctionnalité. La fonction n'est donc pas retirée de la spécification, et il n'est pas prévu de la supprimer dans une future version de RDF.

Une dernière fonctionnalité, mais pas la moins intéressante, permet d'envisager la conciliation du coeur des documents RDF(S) avec des fonctionnalités sémantiques et des extensions définies par d'autres recommandations du W3C depuis 2004 (par exemple, pour les littéraux RDF ordinaires (RDF plain literals), les versions limitées de RDF et RDFS d'implication de régimes définis par SPARQL 1.1, ou POWDER-S IRI Set Semantics). Le but n'est pas de redéfinir ces extensions et fonctionnalités, mais plutôt de les combiner en un seul ou plusieurs document(s) facilement référençables.

3.2. Fonctionnalités prévues

Une fonctionnalité demandée mais qui ne sera pas forcément ajoutée permettrait d'harmoniser l'utilisation de différents types de données (ou leur absence) pour les littéraux ordinaires (ie, les chaînes XSD et littéral ordinaire). L'objectif est de simplifier, par exemple les requêtes SPARQL, qui pour le moment doivent être écrites de trois manières différentes pour obtenir le même résultat. En effet, le problème avec ces littéraux de chaîne est que RDF propose actuellement trois façons différentes de faire quelque chose d'aussi simple que d'écrire une chaîne :

```
"foo" ;
"foo"^^xsd:string ;
"foo@"^^rdf:PlainLiteral. (qui est la façon la
plus bizarre)
```

4. Divers

Mettre à jour et étendre le Primer RDF. Les nouvelles fonctionnalités peuvent inclure plusieurs exemples de syntaxe, qui ne sont plus à jour en terme de vocabulaire utilisé, pouvant traiter de questions autour des données liées comme l'utilisation de « owl:sameAs », l'algorithme de furetage,...

5. Référence

Si vous souhaitez plus de détails sur ces nouveautés consultez la draft RDF 1.1 (Lien 83), Turtle (Lien 84) et JSON (Lien 85).

Retrouvez l'article de Julien Plu en ligne : Lien 86

Les nouveautés de RDFa 1.1

La nouvelle version de RDFa (RDF in attributes) s'oriente plus vers d'autres formats que le XHTML (seule utilisation prévue dans RDFa 1.0) : une spécification est disponible pour XHTML 1.1, mais aussi une autre pour HTML5. Depuis l'utilisation de RDFa dans OpenDocument, ses concepteurs ont pris conscience que RDFa pouvait être utilisé bien ailleurs que ce pour quoi il était prévu.
Cette nouvelle version est prévue pour être aussi compatible que possible avec la 1.0, mais cela n'est pas toujours le cas.

1. Introduction

La nouvelle version de RDFa (RDF in attributes) s'oriente plus vers d'autres formats que le XHTML (seule utilisation prévue dans RDFa 1.0) : une spécification est disponible pour XHTML 1.1, mais aussi une autre pour HTML5. Depuis l'utilisation de RDFa dans OpenDocument, ses concepteurs ont pris conscience que RDFa pouvait être utilisé bien ailleurs que ce pour quoi il était prévu.

Cette nouvelle version est prévue pour être aussi compatible que possible avec la 1.0, mais cela n'est pas toujours le cas.

2. Séparation du Core

Un des plus gros changements est que la spécification est divisée en deux parties : le *Core* et les adaptations aux différents langages (XHTML 1.1 et HTML5). Il n'y a plus une seule spécification, mais bien un ensemble.

Quel avantage ? On peut maintenant adapter RDFa à d'autres langages de manière propre : RDFa ne se limite plus au XHTML, il s'ouvre dès ses premiers pas dans la version 1.1 à une multitude d'usages divers et variés.

3. Vocabulaires par défaut

Il ne sera plus nécessaire de spécifier l'espace de noms de chaque élément : la propriété vocab en définit un par défaut.

En RDFa 1.0, on devait spécifier cet espace de noms :

```
<div xmlns:foaf="http://xmlns.com/foaf/0.1/"
about="#me" rel="foaf:knows">
   <ul>
      <li typeof="foaf:Person"><a
rel="foaf:homepage" property="foaf:name"
href="http://aurore.name/">Aurore</a></li>
      <li typeof="foaf:Person"><a
rel="foaf:homepage" property="foaf:name"
```

```
href="http://johan-is-my-
name.com/">Johan</a></li>
      <li typeof="foaf:Person"><a
rel="foaf:homepage" property="foaf:name"
href="http://isp.net/websites/mywebsitehasnoname/
">Aurélie</a></li>
   </ul>
</div>
```

Avec la 1.1, la propriété vocab remplace ces disgracieuses répétitions :

```
<div vocab="http://xmlns.com/foaf/0.1/"
about="#me" rel=" knows">
   <ul>
      <li typeof="Person"><a rel="homepage"
property="name"
href="http://aurore.name/">Aurore</a></li>
      <li typeof="Person"><a rel="homepage"
property="name" href="http://johan-is-my-
name.com/">Johan</a></li>
      <li typeof="Person"><a rel="homepage"
property="name"
href="http://isp.net/websites/mywebsitehasnoname/
">Aurélie</a></li>
   </ul>
</div>
```

Évidemment, l'effet de cette propriété se limite à ce div, à moins d'être surchargée dans une autre balise.

Ce n'est pas parce qu'on utilise un vocabulaire par défaut qu'on doit se restreindre à ce vocabulaire ! On peut évidemment utiliser d'autres espaces de noms, celui par défaut ne sera utilisé que pour les termes non définis dans ces espaces.

Pour garder une certaine compatibilité avec RDFa 1.0, cette fonctionnalité est cependant à éviter.

4. Documents de profil

Avec ce raccourci, il devient difficile d'utiliser facilement plusieurs vocabulaires. À moins qu'un autre outil ne vienne à la rescousse : les *documents de profil*. Il s'agit de simples documents RDF qui décrivent les correspondances entre les termes et les URI.

Avant, on aurait écrit ceci :

```
<div xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:dc="http://purl.org/dc/elements/1.1/"
about="#me" rel="foaf:knows">
   <ul>
      <li typeof="foaf:Person"><a
rel="foaf:homepage" property="foaf:name"
href="http://aurore.name/">Aurore</a></li>
      <li typeof="foaf:Person"><a
rel="foaf:homepage" property="foaf:name"
href="http://johan-is-my-
name.com/">Johan</a></li>
      <li typeof="foaf:Person"><a
rel="foaf:homepage" property="foaf:name"
href="http://isp.net/websites/mywebsitehasnoname/
">Aurélie</a></li>
   </ul>
   <p>
      Data published by <span
property="dc:creator">me</span>
   </p>
</div>
```

En utilisant un document de profil comme celui-ci, on bénéficiera de raccourcis de syntaxe (ici, il est écrit en Turtle, mais tout autre format de sérialisation convient) :

```
@prefix rdfa: <http://www.w3.org/ns/rdfa#>
[] rdfa:uri  "http://xmlns.com/foaf/0.1/Person";
   rdfa:term "Person" .
[] rdfa:uri
"http://xmlns.com/foaf/0.1/homepage";
   rdfa:term "homepage" .
[] rdfa:uri  "http://xmlns.com/foaf/0.1/name";
   rdfa:term "name" .
[] rdfa:uri
"http://purl.org/dc/elements/1.1/creator";
   rdfa:term "creator" .
```

On peut donc réécrire l'exemple comme ceci :

```
<div profile="http://example.com/profile.ttl"
about="#me" rel="foaf:knows">
   <ul>
      <li typeof="Person"><a rel="homepage"
property="name"
href="http://aurore.name/">Aurore</a></li>
      <li typeof="Person"><a rel="homepage"
property="name" href="http://johan-is-my-
name.com/">Johan</a></li>
      <li typeof="Person"><a rel="homepage"
property="name"
href="http://isp.net/websites/mywebsitehasnoname/
">Aurélie</a></li>
   </ul>
   <p>
      Data published by <span
property="creator">me</span>
   </p>
</div>
```

On peut spécifier plusieurs profils à chaque fois, chaque document de profil peut utiliser toutes les possibilités du format de sérialisation sous-jacent ; à nouveau, si un nouvel ensemble de profils est défini pour une balise enfant, cet ensemble remplacera pour les balises enfant l'ensemble précédemment défini. On ne peut indiquer que des URI à cet endroit.

Pour garder une certaine compatibilité avec RDFa 1.0, cette fonctionnalité est à éviter.

4.1. Comparaison avec les microdonnées (microdata)

Le grand avantage ? RDFa devient plus simple à utiliser, plus court à écrire, surtout si la grande majorité des prédicats utilisés appartient à un seul et même vocabulaire (on peut définir un tel profil directement dans body dans ce cas). Il n'y a plus grande différence avec la version en microdonnées (mais ces microdonnées sont spécifiques à HTML5) - on peut même utiliser plusieurs vocabulaires sans se tracasser. Cette nouvelle version de RDFa veut simplifier les cas les plus simples - aussi les plus fréquents.

```
<div itemscope itemtype="http://www.data-
vocabulary.org/Person/">
   <ul>
      <li><a rel="friend"
href="http://aurore.name/">Aurore</a></li>
      <li><a rel="homepage" href="http://johan-
```

```
is-my-name.com/">Johan</a></li>
          <li><a rel="friend"
href="http://isp.net/websites/mywebsitehasnoname/
">Aurélie</a></li>
   </ul>
</div>
```

On remarque cependant que RDFa est toujours beaucoup plus proche de RDF que les microdonnées.

4.2. Profil par défaut

RDFa 1.1 devrait définir un profil par défaut, soit un petit ensemble de vocabulaires communs, prédéfinis pour chaque document par la spécification. En d'autres termes, il n'y aura pas besoin de définir de préfixe pour ces vocabulaires. Ils ne sont cependant pas encore définis dans la norme. On attend évidemment que ces vocabulaires ne changent pas énormément et qu'ils ne soient pas retirés d'un jour à l'autre de la norme.

5. CURIE

Une autre manière d'indiquer les préfixes est maintenant préférée aux espaces de noms XML : @prefix. En effet, si RDFa est utilisé dans d'autres contextes que ceux pour lesquels il était à l'origine prévu, il est possible qu'il y ait des conflits.

On peut ainsi réécrire le premier exemple :

```
<div prefix="foaf: http://xmlns.com/foaf/0.1/"
about="#me" rel="foaf:knows">
   <ul>
      <li typeof="foaf:Person"><a
rel="foaf:homepage" property="foaf:name"
href="http://aurore.name/">Aurore</a></li>
      <li typeof="foaf:Person"><a
rel="foaf:homepage" property="foaf:name"
href="http://johan-is-my-
name.com/">Johan</a></li>
      <li typeof="foaf:Person"><a
rel="foaf:homepage" property="foaf:name"
href="http://isp.net/websites/mywebsitehasnoname/
">Aurélie</a></li>
   </ul>
</div>
```

Pour garder une compatibilité avec RDFa 1.0, il est conseillé d'utiliser en parallèle les deux mécanismes : définir des préfixes **et** des espaces de noms XML.

Toujours au sujet des CURIE, elles ne sont plus requises partout, elles peuvent maintenant toujours être remplacées au profit d'URI. Cependant, pour garder la compatibilité avec RDFa 1.0, il vaut mieux l'éviter.

6. Références

- RDFa Core 1.1: Syntax and processing rules for embedding RDF through attributes : Lien 87.
- RDFa 1.1 Drafts : Lien 88.
- RDFa 1.1 with a rich snippet example : Lien 89.

Les exemples sont repris en grande partie de Web sémantique : introduction au RDFa (Lien 90).

Retrouvez l'article de Thibaut Cuvelier en ligne : Lien 91

Comment installer la bibliothèque GD - Gestion des dépendances sous Linux et Mac OS

Cet article vous explique comment installer le module GD pour Perl. Il a également pour but de surtout vous expliquer comment installer la bibliothèque GD sous Linux et Mac OS car cela nécessite l'installation de certaines dépendances non évidentes à installer sur un serveur (libpng, libgd…).

1. Introduction

Vous avez sûrement déjà été confronté à devoir installer **GD** pour PHP, Perl, Ruby... Pour certains langages comme PHP, la bibliothèque GD est déjà incorporée, mais pour Perl, Ruby... ce n'est pas le cas : des dépendances doivent être installées.

Pour exemple, essayons d'installer le module Perl GD sous Linux ou Mac OS en **root** sur un serveur n'ayant pas le nécessaire, voici le résultat que l'on obtient :

```
root#perl -MCPAN -e "install GD"
**UNRECOVERABLE ERROR**
Could not find gdlib-config in the search path.
Please install libgd 2.0.28 or higher.
If you want to try to compile anyway, please
rerun this script with the option
--ignore_missing_gd.
Warning: No success on
command[/usr/bin/perl5.10.0 Makefile.PL]
Warning (usually harmless): 'YAML' not installed,
will not store persistent state
  LDS/GD-2.46.tar.gz
  /usr/bin/perl5.10.0 Makefile.PL -- NOT OK
Running make test
  Make had some problems, won't test
Running make install
  Make had some problems, won't install
Could not read '/root/.cpan/build/GD-2.46-
fMf3Dd/META.yml'. Falling back to other methods
to determine prerequisites
```

On constate que l'installation du module est impossible car des dépendances sont nécessaires. Nous verrons dans cet article comment intégrer proprement.

2. Installation de GD sous Windows

Pour installer le module GD sous Windows, il suffit de procéder comme pour tout module Perl en recourant à l'utilitaire **ppm**(**P**erl **P**ackage **M**anager). Il vous importera proprement tout ce qu'il faut. La commande à lancer est la suivante :

```
ppm install GD
```

Pour en savoir plus sur l'installation des modules Perl, lisez cette documentation : Installation des modules Perl CPAN (Lien 92).

3. Installation de GD sous Linux ou Mac OS

Comme je vous l'ai montré dans l'introduction de cet article, sous Linux et Mac OS, l'installation de GD ne fonctionnera probablement pas car il vous manquera certaines dépendances. À moins que vous utilisiez ActivePerl, ce qui est rare mais possible sous Linux/Unix, il va falloir compiler vous-même toutes les dépendances nécessaires. Pas de panique, cet article vous présentera toutes les commandes à lancer pour installer GD. Il va donc falloir recourir aux bibliothèques **zlib**, **libjpeg**, **libpng**, **freetype2** et **libgd**. Puis nous installerons GD par la suite pour Perl.

Pour chaque bibliothèque, je vous mettrai le lien vers le site Web de cette dernière puis un lien vers les sources disponibles dans cet article, car il arrive que les liens Web deviennent obsolètes après plusieurs mois ou années. Pour pallier cela, les sources ont été mises sur le domaine de developpez.com.

3.1. Prérequis

Sans vouloir insister car ce n'est pas le but de cet article, toute installation et compilation sur une machine sous Linux/Mac OS nécessite d'avoir au préalable les utilitaires **gcc**, **make**… installés.

3.2. zlib

zlib est une bibliothèque logicielle de compression de données. Elle implémente l'algorithme de compression *deflate* et peut créer des fichiers au format *gzip*.

Nous pouvons trouver toutes les versions de zlib sur le site officiel : Lien 93. À ce jour, la version la plus récente est la version 1.2.6. Une copie est disponible ici : Lien 94.

Voici les commandes d'installation sous root, de préférence dans le répertoire temporaire de notre système.

```
mkdir -p /tmp/GD
cd /tmp/GD
wget
http://djibril.developpez.com/tutoriels/perl/comment-installer-bibliotheque-gd/fichiers/zlib-1.2.6.tar.gz
tar -xzvf ./zlib-1.2.6.tar.gz
cd zlib-1.2.6
./configure --shared && make && make install
```

3.3. libjpeg

Le paquet **libjpeg** contient les bibliothèques *JPEG*. Elles permettent la compression de fichiers images basés sur le standard JPEG (Joint Photographic Experts Group). Il s'agit d'un algorithme, avec perte de qualité.

Nous pouvons trouver toutes les versions de JPEG sur le site : Lien 95. Une copie est disponible ici : Lien 96.

Sous Linux

```
mkdir -p /tmp/GD
cd /tmp/GD
wget
http://djibril.developpez.com/tutoriels/perl/comment-installer-bibliotheque-gd/fichiers/jpegsrc.v6b.tar.gz
tar -xzvf ./jpegsrc.v6b.tar.gz
cd jpeg-6b
./configure --enable-shared && make && make install
```

Sous Mac OS

```
mkdir -p /tmp/GD
cd /tmp/GD
wget
http://djibril.developpez.com/tutoriels/perl/comment-installer-bibliotheque-gd/fichiers/jpegsrc.v6b.tar.gz
tar -xzvf ./jpegsrc.v6b.tar.gz
cd jpeg-6b
ln -s `which glibtool` ./libtool
export MACOSX_DEPLOYMENT_TARGET=10.4
./configure --enable-shared && make && make install
```

3.4. libpng

Le paquet **libpng** contient des bibliothèques utilisées par d'autres programmes pour lire et écrire des fichiers PNG.

Nous pouvons trouver toutes les versions de libpng sur le site : Lien 97. Une copie est disponible ici : Lien 98.

```
mkdir -p /tmp/GD
cd /tmp/GD
wget
http://djibril.developpez.com/tutoriels/perl/comment-installer-bibliotheque-gd/fichiers/libpng-1.2.48.tar.gz
tar -xzvf ./libpng-1.2.48.tar.gz && cd libpng-1.2.48
./configure && make && make install
```

3.5. freetype2

Cette bibliothèque vous permet d'utiliser des polices TrueType sous Linux.

Nous pouvons trouver toutes les versions de libpng sur le site : Lien 99. Une copie est disponible ici : Lien 100.

```
mkdir -p /tmp/GD
cd /tmp/GD
wget
http://djibril.developpez.com/tutoriels/perl/comment-installer-bibliotheque-gd/fichiers/freetype-2.4.9.tar.gz
tar -xzvf freetype-2.4.9.tar.gz  && cd freetype-2.4.9
./configure && make && make install
```

3.6. libgd

libgd est le nom d'une bibliothèque libre servant à manipuler des images dynamiquement.

Nous pouvons trouver toutes les versions de libgd sur le site : Lien 101. Une copie est disponible ici : Lien 102.

```
mkdir -p /tmp/GD
cd /tmp/GD
wget
http://djibril.developpez.com/tutoriels/perl/comment-installer-bibliotheque-gd/fichiers/pierrejoye-gd-libgd-5551f61978e3.tar.gz
tar -xvzf pierrejoye-gd-libgd-5551f61978e3.tar.gz
cd pierrejoye-gd-libgd-5551f61978e3/src
ln -s /usr/X11R6/include/fontconfig /usr/local/include
./configure && make && make install
```

3.7. GD

Maintenant que toutes les bibliothèques nécessaires sont intégrées, nous pouvons installer notre module Perl **GD** en toute tranquillité de la sorte :

```
perl -MCPAN -e "install GD"
perl -MCPAN -e "install GD::Text"
```

À ce stade, les bibliothèques et le module GD sont installés et à jour. Vous pouvez maintenant utiliser et installer comme bon vous semble tout module Perl nécessitant GD et exécuter vos programmes.

3.8. Résumé

Pour vous simplifier la vie, voici un programme complet qui regroupe toutes les commandes de cet article pour une installation d'une traite sous l'utilisateur **root** !

Script complet d'installation

```
#!/bin/bash
#================================================================================
# Auteur : djibril
# Date   : 25/03/2012 19:37:25
# But    : Installation des bibliotheques libpng, libjpeg... et des modules Perl GD
#================================================================================

echo "============================";
echo "Creation d'un repertoire temporaire pour les installations";
echo "============================";
mkdir -p /tmp/GD

echo "============================";
echo "Installation de zlib";
echo "============================";
cd /tmp/GD
wget
http://djibril.developpez.com/tutoriels/perl/comment-installer-bibliotheque-gd/fichiers/zlib-1.2.6.tar.gz
tar -xzvf ./zlib-1.2.6.tar.gz
cd zlib-1.2.6
./configure --shared && make && make install


echo "============================";
echo "Installation de libjpeg";
echo "============================";
cd /tmp/GD
```

```
wget
http://djibril.developpez.com/tutoriels/perl/comment-installer-bibliotheque-gd/fichiers/jpegsrc.v6b.tar.gz
tar -xzvf ./jpegsrc.v6b.tar.gz
cd jpeg-6b
./configure --enable-shared && make && make install

echo "============================";
echo "Installation de libpng";
echo "============================";
cd /tmp/GD
wget
http://djibril.developpez.com/tutoriels/perl/comment-installer-bibliotheque-gd/fichiers/libpng-1.2.48.tar.gz
tar -xzvf ./libpng-1.2.48.tar.gz && cd libpng-1.2.48
./configure && make && make install


echo "============================";
echo "Installation de freetype2";
echo "============================";
cd /tmp/GD
wget
http://djibril.developpez.com/tutoriels/perl/comment-installer-bibliotheque-gd/fichiers/freetype-2.4.9.tar.gz
tar -xzvf freetype-2.4.9.tar.gz  && cd freetype-2.4.9
./configure && make && make install

echo "============================";
echo "Installation de libgd";
echo "============================";
cd /tmp/GD
wget
http://djibril.developpez.com/tutoriels/perl/comment-installer-bibliotheque-gd/fichiers/pierrejoye-gd-libgd-5551f61978e3.tar.gz
tar -xvzf pierrejoye-gd-libgd-5551f61978e3.tar.gz
cd pierrejoye-gd-libgd-5551f61978e3/src
ln -s /usr/X11R6/include/fontconfig /usr/local/include
./configure && make && make install

echo "Installation du module Perl GD et GD::Text";
perl -MCPAN -e "install GD"
perl -MCPAN -e "install GD::Text"
```

Vous devez enregistrer ce programme dans un répertoire et lancer sous root le fichier de la façon suivante :

```
mkdir -p /tmp/GD && cd /tmp/GD
wget
http://djibril.developpez.com/tutoriels/perl/comment-installer-bibliotheque-gd/fichiers/installation-complete.sh && sh installation-complete.sh
```

4. Conclusion

J'espère que cette documentation vous a aidé à installer facilement cette bibliothèque. Maintenant, il vous reste votre imagination pour concevoir des programmes révolutionnaires :-) !

Retrouvez l'article de djibril en ligne : Lien 103

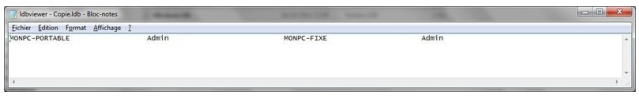
Les fichiers de verrouillage Access (ldb et laccdb)

Pour gérer les connexions aux fichiers de base de données, Access utilise des fichiers de verrouillage (extension ldb ou laccdb).
Nous découvrirons dans cet article comment sont créés et remplis ces fichiers.
Nous verrons également comment lire les verrous posés sur ces fichiers pour connaître les connexions actives.

1. Introduction

Vous avez sans doute remarqué, à l'ouverture d'une base de données, qu'un fichier avec l'extension **ldb** ou **laccdb** est généralement créé.

Ce fichier contient une liste d'utilisateurs (et le nom de leur ordinateur) qui se sont connectés à la base de données.



Mais cette liste n'est pas utilisable en l'état sans connaître le mécanisme de verrous utilisé.

Ce mécanisme est expliqué dans un document (en anglais) que vous pouvez trouver ici :
ACC : Utilitaires de Microsoft Jet disponibles dans le centre de téléchargement (Lien 104).

Le document en question est **Jetlock.doc** *(Understanding Microsoft Jet Locking White Paper)*.
Il contient beaucoup d'informations techniques pas forcément à jour pour les dernières versions d'Access.
Il est donc réservé aux plus curieux et motivés !

Cet article explique ce mécanisme de connexion aux bases de données et comment lire les informations des connexions actives.
En fin d'article, vous pourrez télécharger le code qui permet de dresser la liste des utilisateurs d'une base de données.

2. Le fichier de verrouillage

Le fichier de verrouillage est géré par le moteur de base de données :
- **JET** jusqu'à Access 2003 inclus ;
- **ACE** à partir d'Access 2007.

Ce n'est donc pas l'application Access qui crée ce fichier.
Un accès à la base de données par une application externe utilisera donc ce même mécanisme de verrous.

2.1. Extension du fichier

Le fichier de verrouillage a une extension **ldb** pour les formats de fichier jusqu'à Access 2003 (mdb = **JET**).

À partir du format de fichier Access 2007 (accdb = **ACE**), le fichier de verrouillage a une extension **laccdb**.

2.2. Structure

Il contient une liste d'ordinateurs et d'utilisateurs.
Chacune de ces deux informations est écrite sur une longueur fixe de 32 caractères.
Un caractère Null (chr(0) ou vbNullChar) marque la fin de chaque information.

Si on ouvre le fichier de verrouillage avec Notepad++ par exemple (Lien 105), on voit très bien ce caractère de fin **NUL** :

Le nom d'utilisateur est celui de la sécurité au niveau utilisateur.
Si aucune sécurité utilisateur n'est mise en place, Admin sera l'utilisateur pour toutes les connexions.
Seul le nom de PC pourra alors différencier les utilisateurs.

Le nombre maximal de connexions est 255.
Le fichier de verrouillage sera donc d'une taille maximale de 255 * 64 = 16 320 octets.

2.3. Création, Modification et Suppression

Le fichier de verrouillage est créé à la première connexion (partagée) à la base de données.
Ce fichier est placé dans le même répertoire que le fichier de la base de données et porte le même nom, à part l'extension qui diffère.

Lors d'une ouverture de la base de données en mode exclusif, aucun fichier de verrouillage n'est créé.
C'est le fichier de base de données (mdb, accdb...) qui est verrouillé. Une seule personne à la fois peut ouvrir une base de données en mode exclusif.

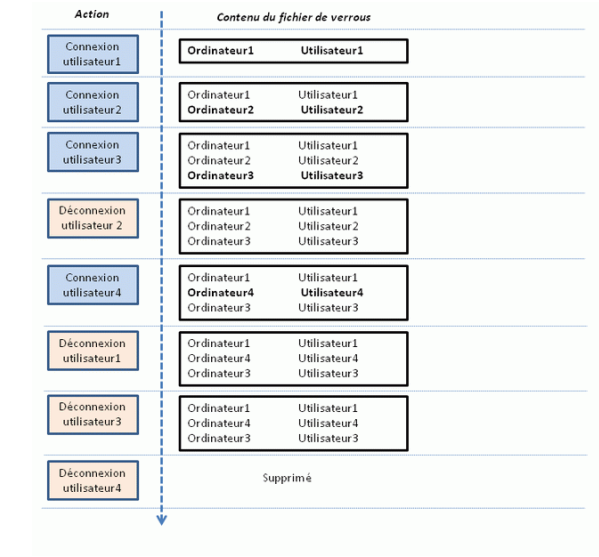
Dans ce fichier, une ligne est ajoutée (ou modifiée) avec les informations de l'utilisateur (cf. le chapitre suivant pour la structure des données).

Lors de nouvelles connexions, une ligne est ajoutée pour chaque nouvel utilisateur.

Attention : lors d'une déconnexion, la ligne correspondant à l'utilisateur n'est pas supprimée.

La ligne d'information d'un utilisateur déconnecté est marquée comme disponible grâce à un mécanisme de verrou que nous verrons dans le chapitre suivant.
Une prochaine connexion écrasera les informations d'une ligne disponible dans le fichier s'il en existe une, ou créera une nouvelle ligne si nécessaire.

Lorsqu'il n'y a plus aucun utilisateur connecté, le fichier de verrou est supprimé.



Notez que le contenu du fichier de verrous est présenté avec une ligne par connexion pour une meilleure compréhension.
En réalité les informations sont écrites dans le fichier sans retour à la ligne.

Lire le contenu du fichier de verrouillage n'est donc pas suffisant pour connaître la liste des utilisateurs connectés.
Des utilisateurs qui se sont connectés puis déconnectés peuvent apparaître dans la liste.

2.4. Les Verrous

2.4.1. Le verrouillage de fichiers sous Windows

Un fichier sous Windows peut être ouvert en mode partagé ou en mode exclusif :
- en mode partagé, chaque utilisateur peut verrouiller une partie différente du fichier ;
- en mode exclusif, l'ensemble du fichier est verrouillé et un seul utilisateur à la fois peut ouvrir le fichier.

Dans ces deux cas, le fichier verrouillé en totalité ou en partie ne peut pas être supprimé.

Type d'ouverture du fichier	Type de verrouillage	Utilisateurs simultanés
Partagé	Chaque utilisateur peut verrouiller une partie du fichier	Plusieurs possibles
Exclusif	Un seul utilisateur verrouille l'ensemble du fichier	Un seul

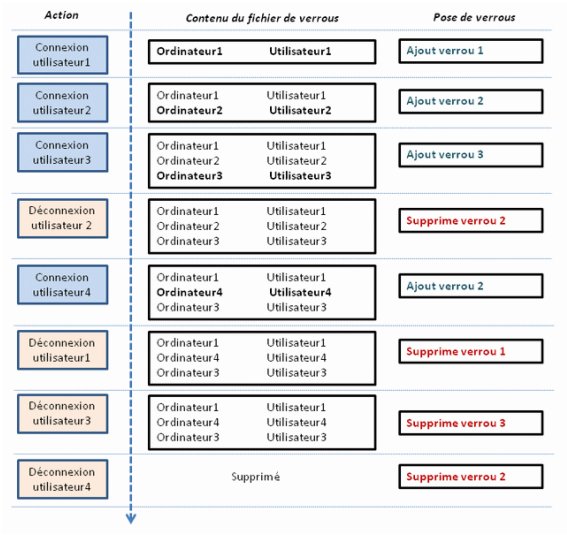
Les bases de données Access sont ouvertes en mode exclusif ou partagé.

Les fichiers de verrouillage Access sont ouverts en mode partagé.

2.4.2. Verrouillage des fichiers ldb et accdb

À chaque connexion, le système vérifie si une connexion est disponible.
Un verrou est alors posé pour cette connexion.

Lorsqu'un utilisateur quitte la base de données, le verrou utilisé est alors supprimé et la connexion est réutilisable.



On voit sur ce diagramme que la connexion de l'utilisateur 2 est réutilisée par l'utilisateur 4.
Pourtant la deuxième "ligne" du fichier était déjà renseignée avec les données de l'utilisateur 2, mais le verrou (qui n'est pas visible) a été supprimé.

3. Lire et filtrer le fichier de verrouillage

Dans un premier temps nous allons lire le contenu d'un fichier de verrouillage.
Ensuite nous verrons comment filtrer la liste des utilisateurs pour ne garder que ceux qui sont connectés.

3.1. Lire le contenu du fichier

On pourrait lire le fichier de verrouillage grâce aux fonctions VBA.
Voici un exemple de code pour illustrer cette éventualité.

```
Sub ReadLDB()
Dim iFile As Integer
Dim sComputer As String
Dim sUser As String
Dim cptUser As Long
iFile = FreeFile
Open "C:\MaBase.ldb" For Input As iFile
For cptUser = 1 To LOF(iFile) / 64
    sComputer = RTrim(Input(31, iFile))
    sUser = RTrim(Input(31, iFile))
    Debug.Print sComputer & ":" & sUser
Next
Close iFile
End Function
```

Chaque entrée dans le fichier contient l'ordinateur et l'utilisateur, chacun sur trente-deux caractères.
Notez qu'on ne lit que trente-et-un caractères. Le trente-deuxième est un caractère Null que l'instruction **Input** ignore.

Lien vers un tutoriel : Manipulation des fichiers en VBA (Lien 106)

Nous allons plutôt utiliser des API pour lire le fichier, car elles nous seront également utiles pour lire les verrous.
Ces API sont des fonctions contenues dans la bibliothèque **kernel32.dll**.

Voici leurs déclarations, ainsi que les constantes dont nous avons besoin :

Déclaration des API et constantes

```
Private Declare Function ReadFile Lib "kernel32"
(ByVal hFile As Long, lpBuffer As Any, _
                ByVal nNumberOfBytesToRead As
Long, lpNumberOfBytesRead As Long, lpOverlapped
As Any) As Long
Private Declare Function CreateFile Lib
"kernel32" Alias "CreateFileA" (ByVal lpFileName
As String, _
                ByVal dwDesiredAccess As Long,
ByVal dwShareMode As Long, lpSecurityAttributes
As Any, _
                ByVal dwCreationDisposition As
Long, ByVal dwFlagsAndAttributes As Long, _
                ByVal hTemplateFile As Long) As
Long
Private Declare Function CloseHandle Lib
"kernel32" (ByVal hObject As Long) As Long
Private Declare Function GetFileSize Lib
"kernel32" (ByVal hFile As Long, lpFileSizeHigh
As Long) As Long
Private Declare Function LockFile Lib "kernel32"
(ByVal hFile As Long, ByVal dwFileOffsetLow As
Long, _
                 ByVal dwFileOffsetHigh As Long,
ByVal nNumberOfBytesToLockLow As Long, _
                ByVal nNumberOfBytesToLockHigh As
Long) As Long
Private Declare Function UnlockFile Lib
"kernel32" (ByVal hFile As Long, ByVal
dwFileOffsetLow As Long, _
                ByVal dwFileOffsetHigh As Long,
ByVal nNumberOfBytesToUnlockLow As Long, _
                ByVal nNumberOfBytesToUnlockHigh
As Long) As Long
Private Declare Function SetFilePointer Lib
"kernel32" (ByVal hFile As Long, ByVal
lDistanceToMove As Long, _
                lpDistanceToMoveHigh As Long,
ByVal dwMoveMethod As Long) As Long

Private Const GENERIC_READ = &H80000000      '
Ouverture en lecture
Private Const GENERIC_WRITE = &H40000000     '
Ouverture en écriture
Private Const FILE_SHARE_READ = &H1          '
Lecture partagée
Private Const FILE_SHARE_WRITE = &H2         '
Écriture partagée
Private Const OPEN_EXISTING = 3              ' Le
fichier doit exister
Private Const DEBUT_LOCK = &H10000001        '
Adresse de début des locks
```

Pour simplifier, le code est limité aux versions 32 bits d'Office.
Pour une version 64 bits d'Office, voir cet article : Développer avec Office 64 bits (Lien 107).

Les modules de l'application à télécharger sont compatibles avec Office 64 bits : Lien 108.

Retrouvez la documentation de ces fonctions sur MSDN : File Management Functions (Lien 109).

Voici une procédure qui utilise les API déclarées précédemment pour lire le contenu du fichier de verrous :

```
Sub ReadLDB()
Dim hFile As Long
Dim lReturn As Long
Dim lBytesRead As Long
Dim sComputer As String
Dim sUser As String
' Ouverture fichier en mode partagé
hFile = CreateFile(ByVal
"C:\Article_Dvp\documents\utilisateurs-
ldb\work\ldbviewer - Copie.ldb", _
                            ByVal GENERIC_READ Or
GENERIC_WRITE, _
                            ByVal FILE_SHARE_READ
Or FILE_SHARE_WRITE, _
                            ByVal 0&, ByVal
OPEN_EXISTING, ByVal 0&, ByVal 0&)
' Si -1 => erreur
If hFile <> -1 Then
    ' Boucle pour lire le contenu du fichier
    Do
        ' Initialise la zone qui reçoit le nom de
l'ordinateur
        sComputer = Space(32)
        ' Lecture de 32 caractères
        lReturn = ReadFile(hFile, ByVal
sComputer, 32, lBytesRead, ByVal 0&)
        ' Si erreur ou plus de données => on sort
        If lReturn = 0 Or lBytesRead = 0 Then
Exit Do
        ' Retire le Null et les espaces à droite
        sComputer = Left(sComputer,
InStr(sComputer, Chr(0)) - 1)
        ' Initialise la zone qui reçoit le nom de
l'utilisateur
        sUser = Space(32)
        ' Lecture de 32 caractères
        lReturn = ReadFile(hFile, ByVal sUser,
32, lBytesRead, ByVal 0&)
        ' Retire le Null et les espaces à droite
        sUser = Left(sUser, InStr(sUser, Chr(0))
- 1)
        ' Si erreur ou plus de données => on sort
        If lReturn = 0 Or lBytesRead = 0 Then
Exit Do
        ' Écriture dans fenêtre exécution
        Debug.Print sComputer & ":" & sUser
    Loop
    ' Fermeture du fichier
    CloseHandle hFile
End If
End Function
```

Il nous reste à filtrer les connexions non actives.

3.2. Filtrer le contenu du fichier grâce aux verrous

Les verrous sont posés par Access à un emplacement

défini à &H10000001 (en hexadécimal), c'est-à-dire au-delà du fichier physique.
Cf. Understanding Microsoft Jet Locking White Paper pour plus d'explication (en anglais) : Lien 110.

Cet emplacement a été défini précédemment par la constante **DEBUT_LOCK**.
La première connexion a donc son verrou à DEBUT_LOCK.
La deuxième connexion à DEBUT_LOCK + 1.
Il peut y avoir 255 connexions au maximum.

Windows peut placer un verrou à une position supérieure à la taille de fichier.
Cela ne verrouillera pas un endroit spécifique du fichier.
Imaginez un catalogue de verrous liés à un fichier, et non pas un verrou posé "sur" ce fichier.

En fait il n'est pas possible de lire l'état d'un verrou.
Il faut tenter d'en poser un : si cela retourne une erreur c'est qu'il y a déjà un verrou.

Voici la procédure avec l'ajout de la tentative de pose de verrou pour savoir si la connexion est active.

```
Function ReadLDB()
Dim hFile As Long
Dim lReturn As Long
Dim lBytesRead As Long
Dim sComputer As String
Dim sUser As String
Dim lNbUser As Long
' Ouverture fichier en mode partagé
hFile = CreateFile(ByVal
"C:\Article_Dvp\documents\utilisateurs-
ldb\work\ldbviewer.ldb", _
                           ByVal GENERIC_READ Or
GENERIC_WRITE, _
                           ByVal FILE_SHARE_READ
Or FILE_SHARE_WRITE, _
                           ByVal 0&, ByVal
OPEN_EXISTING, ByVal 0&, ByVal 0&)
' Si -1 => erreur
If hFile <> -1 Then
    ' Boucle pour lire le contenu du fichier
    Do
        lNbUser = lNbUser + 1
        ' Initialise la zone qui reçoit le nom de
l'ordinateur
        sComputer = Space(32)
        ' Lecture de 32 caractères
        lReturn = ReadFile(hFile, ByVal
sComputer, 32, lBytesRead, ByVal 0&)
        ' Si erreur ou plus de données => on sort
        If lReturn = 0 Or lBytesRead = 0 Then
Exit Do
        ' Retire le Null et les espaces à droite
        sComputer = Left(sComputer,
InStr(sComputer, Chr(0)) - 1)
        ' Initialise la zone qui reçoit le nom de
l'utilisateur
        sUser = Space(32)
        ' Lecture de 32 caractères
        lReturn = ReadFile(hFile, ByVal sUser,
32, lBytesRead, ByVal 0&)
        ' Retire le Null et les espaces à droite
        sUser = Left(sUser, InStr(sUser, Chr(0))
- 1)
        ' Si erreur ou plus de données => on sort
        If lReturn = 0 Or lBytesRead = 0 Then
Exit Do
        ' Test le lock pour savoir si
l'utilisateur est toujours connecté
        If LockFile(hFile, DEBUT_LOCK + lNbUser -
1, 0, 1, 0) = 0 Then
            ' Erreur de lock => utilisateur
connecté
            ' Écriture dans fenêtre exécution
            Debug.Print sComputer & ":" & sUser
        Else
            ' Lock réussi => on "unlock"
            Call UnlockFile(hFile, DEBUT_LOCK +
lNbUser - 1, 0, 1, 1)
        End If
    Loop
    ' Fermeture du fichier
    CloseHandle hFile
End If
End Function
```

On a dû rajouter un compteur **lNbUser** qui permet d'incrémenter la position du verrou à tester.

Ne pas oublier de déverrouiller si la tentative de pose de verrou aboutit.
Cependant, les verrous posés sont supprimés lorsque le fichier **hFile** est fermé par **CloseHandle**.

4. Cas particulier du mode exclusif

Si la base est ouverte en mode exclusif, aucun fichier de verrouillage n'est créé.
Il est possible de détecter qu'une base est ouverte en mode exclusif en essayant de l'ouvrir... en mode exclusif.

```
Function isExclusive(pDataBase As String) As
Boolean
Dim f As Integer
On Error GoTo Gestion_Erreur
' Cherche un identifiant fichier libre
f = FreeFile
' Tente d'ouvrir la base de données en lecture
exclusive
Open pDataBase For Binary Access Read Lock Read
As #f
' Ferme le fichier s'il a été ouvert
Close f
Exit Function
Gestion_Erreur:
' Erreur 70 (Permission refusée) si base déjà
ouverte en exclusif
If Err.Number = 70 Then isExclusive = True
End Function
```

Dans la fonction ci-dessus, l'instruction **Open** lève une erreur si la base (dont le chemin est donné dans le paramètre **pDatabase**) est déjà ouverte en mode exclusif.

Il est possible de faire la même tentative d'ouverture avec les API : cf. les modules de l'application à télécharger pour un exemple (Lien 108).

5. Cas de la fermeture brutale d'Access

Ce mécanisme de verrou a un avantage non négligeable : le verrou ne survit pas à la fermeture du processus (donc de l'application) qui l'a posé.

C'est-à-dire que la connexion est rendue disponible car il n'y a plus de verrou dessus.

Notez que le fichier ldb (ou accdb) n'est pas supprimé lors de la déconnexion brutale du dernier utilisateur.
Comme il n'y a plus de verrou sur le fichier, il est possible (mais pas indispensable) de le supprimer manuellement.

6. Verrouiller les connexions

6.1. En posant des verrous sur le fichier ldb ou accdb

Ce paragraphe et notamment le code qui s'y trouve n'est pas prévu pour fonctionner en production.
Ce n'est qu'un exemple pour illustrer le mécanisme de verrou.

Petite astuce si vous souhaitez empêcher toute nouvelle connexion à la base de données :
Posez un verrou avec **LockFile** sur chacune des 255 connexions.
Cela nécessite de conserver un pointeur vers le fichier (le **hFile** obtenu avec **CreateFile**) pour que les verrous perdurent.

Voici un module pour exemple :

```
Option Compare Database
Option Explicit

Private Declare Function ReadFile Lib "kernel32" (ByVal hFile As Long, lpBuffer As Any, _
                ByVal nNumberOfBytesToRead As Long, lpNumberOfBytesRead As Long, lpOverlapped As Any) As Long
Private Declare Function CreateFile Lib "kernel32" Alias "CreateFileA" (ByVal lpFileName As String, _
                ByVal dwDesiredAccess As Long, ByVal dwShareMode As Long, lpSecurityAttributes As Any, _
                ByVal dwCreationDisposition As Long, ByVal dwFlagsAndAttributes As Long, _
                ByVal hTemplateFile As Long) As Long
Private Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As Long
Private Declare Function GetFileSize Lib "kernel32" (ByVal hFile As Long, lpFileSizeHigh As Long) As Long
Private Declare Function LockFile Lib "kernel32" (ByVal hFile As Long, ByVal dwFileOffsetLow As Long, _
                 ByVal dwFileOffsetHigh As Long, ByVal nNumberOfBytesToLockLow As Long, _
                ByVal nNumberOfBytesToLockHigh As Long) As Long
Private Declare Function UnlockFile Lib "kernel32" (ByVal hFile As Long, ByVal dwFileOffsetLow As Long, _
                ByVal dwFileOffsetHigh As Long, ByVal nNumberOfBytesToUnlockLow As Long, _
                ByVal nNumberOfBytesToUnlockHigh As Long) As Long
Private Declare Function SetFilePointer Lib "kernel32" (ByVal hFile As Long, ByVal lDistanceToMove As Long, _
                lpDistanceToMoveHigh As Long, ByVal dwMoveMethod As Long) As Long

Private Const GENERIC_READ = &H80000000      '
Ouverture en lecture
Private Const GENERIC_WRITE = &H40000000     '
Ouverture en écriture
Private Const FILE_SHARE_READ = &H1          '
Lecture partagée
Private Const FILE_SHARE_WRITE = &H2         '
Écriture partagée
Private Const OPEN_EXISTING = 3              ' Le
fichier doit exister
Private Const DEBUT_LOCK = &H10000001        '
Adresse de début des locks

Private hFile As Long

Public Sub LockDb(pFile As String)
Dim lNbUser As Long
Dim lNbOK As Long
Dim lNbKO As Long
' Ouverture fichier en mode partagé
hFile = CreateFile(ByVal pFile, _
                            ByVal GENERIC_READ Or
GENERIC_WRITE, _
                            ByVal FILE_SHARE_READ
Or FILE_SHARE_WRITE, _
                            ByVal 0&, ByVal
OPEN_EXISTING, ByVal 0&, ByVal 0&)
' Si -1 => erreur
If hFile <> -1 Then
    ' Boucle sur les 255 verrous
    For lNbUser = 1 To 255
        ' Pose d'un verrou
        If LockFile(hFile, DEBUT_LOCK + lNbUser -
1, 0, 1, 0) = 0 Then
            ' Connexion déjà utilisée => pose de
verrou impossible
            lNbKO = lNbKO + 1
        Else
            ' Pose de verrou réussie
            lNbOK = lNbOK + 1
        End If
    Next
End If
MsgBox lNbOK & " connexions verrouillées" &
vbCrLf & _
        lNbKO & " connexions en utilisation"
End Sub

Public Sub UnlockDb()
Dim lNbUser As Long
Dim lNbOK As Long
Dim lNbKO As Long
' Si -1 => erreur
If hFile <> -1 And hFile <> 0 Then
    ' Boucle sur les 255 verrous
    For lNbUser = 1 To 255
        ' Retire un verrou
        If UnlockFile(hFile, DEBUT_LOCK + lNbUser
- 1, 0, 1, 1) = 0 Then
            ' Connexion déjà utilisée par un
autre process => retrait de verrou impossible
            lNbKO = lNbKO + 1
        Else
            ' Retrait de verrou réussi
            lNbOK = lNbOK + 1
        End If
    Next
End If
' Libère le fichier
CloseHandle hFile
hFile = 0
MsgBox lNbOK & " connexions déverrouillées" &
```

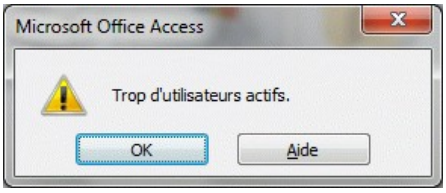
```
vbCrLf & _
        lNbKO & " connexions en utilisation"
End Sub
```

Verrouillez les connexions avec la procédure **LockDb** :

```
LockDb "C:\MaBase.ldb"
```

Mettez en paramètre le fichier ldb ou accdb.
Notez que seules les connexions libres peuvent être verrouillées.

Après verrouillage de toutes les connexions, un message apparait à l'ouverture de la base de données comme s'il y avait 255 utilisateurs connectés.



Exécutez la procédure **UnLockDb** pour déverrouiller les connexions.

Il faut un fichier ldb ou accdb pour verrouiller une base de données.
Si aucun utilisateur n'est connecté, on peut créer un fichier texte vide nommé *NomDeLaBase.ldb* ou *NomDeLaBase.accdb* et y poser des verrous.

Les verrous posés sont retirés lorsqu'on ferme l'application qui les a créés, même si on n'exécute pas la fonction **UnlockFile**.

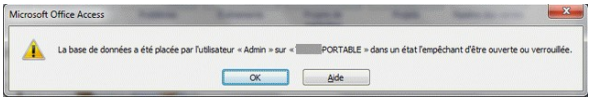
6.2. Avec ADO et Connection Control

Beaucoup plus simple (à partir d'**Access 2000**), ouvrez la base de données à verrouiller et exécutez ce code :

```
CurrentProject.Connection.Properties("Jet
OLEDB:Connection Control") = 1
```

La base de données est alors verrouillée pour les nouvelles connexions, tant que vous ne fermez pas la base de données d'où vous avez exécuté ce code.
Les utilisateurs déjà connectés conservent leur connexion jusqu'à sa fermeture.

Un message apparait à l'ouverture de la base de données :



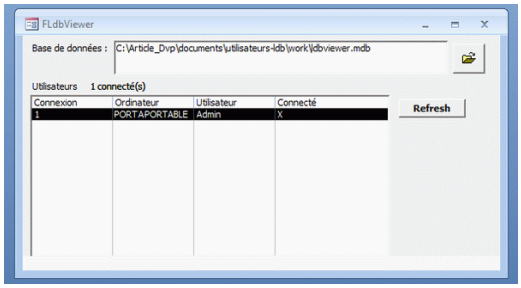
Pour déverrouiller la base de données, exécutez le même code avec la valeur 2 au lieu de 1.

Source : Connection Control (Lien 111).

7. Application à télécharger

Voici une application qui utilise la technique expliquée dans cet article.
clLDBUser est un module qui contient les données des utilisateurs.
clLDBViewer est le module qui contient le code permettant de lister les utilisateurs connectés.

Le formulaire **FLdbViewer** utilise ces deux modules au sein d'une interface.



Vous pouvez réutiliser les modules de code **clLDBUser** et **clLDBViewer**.

Télécharger l'application au format Access 2000 : Lien 112.

8. Autres possibilités pour lister les utilisateurs

La méthode **OpenSchema** de la bibliothèque ADO permet de lire les connexions d'une base de données.
Lien MSDN : Use ADO to Return a List of Users Connected to a Database (Lien 113).

Si vous avez besoin d'un outil prêt à l'emploi, utilisez cette visionneuse d'argyronet : Visionneuse des Utilisateurs d'une base de données Access (Lien 114).

Retrouvez l'article de Thierry Gasperment en ligne : Lien 115.

Qu'est-ce que l'informatique décisionnelle ?

Ce tutoriel, destiné aux débutants, a pour objectif de vous expliquer ce qu'est l'informatique décisionnelle. Cet article a été initialement publié sur mon blog (Lien 116) et est extrait du rapport TEST « Capacités des outils SOLAP en termes de requêtes spatiales, temporelles et spatio-temporelles » que j'ai rédigé en vue de l'obtention de l'unité ENG111 du CNAM, en juin 2011.

1. Introduction

Conscients que l'une des plus grandes richesses d'une entreprise est son information, mais noyés sous de nombreuses données, éparses, déstructurées et hétérogènes, les dirigeants sont face à une problématique de taille : comment analyser ces informations, dans des temps raisonnables ? Celles-ci concernent-elles toutes les mêmes périodes ? Ces décideurs ont besoin qu'on leur expose les faits importants, base de leurs décisions.

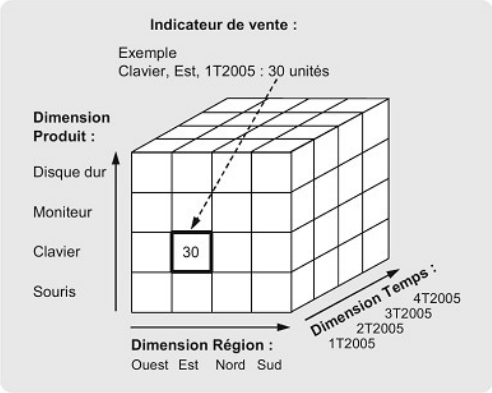
C'est ce à quoi l'**informatique décisionnelle** (aussi nommée DSS pour *Decision Support System* ou encore *BI pour Business Intelligence*) est destinée. Elle prend une place en constante croissance dans les systèmes d'information (SI) depuis son apparition, dans les années **quatre-vingt-dix**.

Elle permet une sélection des informations opérationnelles pertinentes pour l'entreprise. Celles-ci sont ensuite normalisées pour alimenter un entrepôt de données. De ce concept est née la notion de modélisation dimensionnelle. Cette dernière est fondamentale pour répondre aux exigences de rapidité et de facilité d'analyse. Elle permet, en outre, de rendre les données d'un entrepôt cohérentes, lisibles, intelligibles et faciles d'accès.

L'informatique décisionnelle doit produire des indicateurs et des rapports à l'attention des analystes. Elle doit également proposer des outils de navigation, d'interrogation et de visualisation de l'entrepôt.

2. Le modèle multidimensionnel

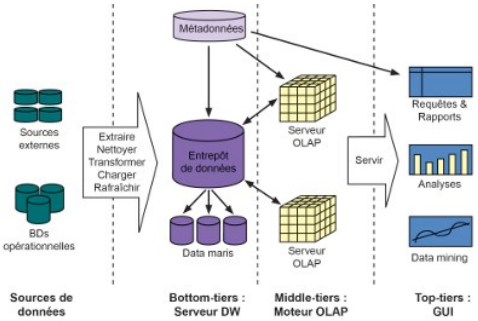
Le modèle multidimensionnel est la combinaison de tables de dimensions et de faits. Le fait est le sujet de l'analyse. Il est formé de mesures, généralement numériques, renseignées de manière continue. Ces mesures permettent de résumer un grand nombre d'enregistrements des données sources en quelques-uns. Le fait est analysé selon des perspectives, nommées dimensions. Chacune contient une structure hiérarchique ; la dimension « temps », par exemple, pourrait être divisée en années, trimestres, mois, semaines, jours...



Cube multidimensionnel à trois perspectives d'analyse (inspiré de 'Introduction pratique aux bases de données relationnelles')

De cette hiérarchie découle le niveau de granularité de l'entrepôt, et donc, les niveaux d'agrégations. La figure ci-dessus montre le cube permettant l'analyse de l'« indicateur de vente » selon trois dimensions : produit, temps (divisé en trimestres), et région.

3. Architecture d'un système décisionnel



Ensemble des composants intervenant dans un système décisionnel (Source: 'Informatique Décisionnelle' NFE115)

Cette section propose de parcourir les différents éléments nécessaires à la mise en place d'une solution d'aide à la décision, depuis l'extraction des données jusqu'à leur restitution sous forme agrégée, synthétisée et normalisée.

3.1. Sources de données

Afin d'alimenter les entrepôts, les informations doivent être identifiées et extraites de leurs emplacements originels. Il s'agit majoritairement de données internes à l'entreprise, mais diffuses, car stockées dans les bases de données de production des différents services (legacy

systems). Ce peut être aussi des sources externes, récupérées via des services distants, des web services, par exemple. Ce sont des données complexes : plusieurs technologies (types de fichiers, encodages, liens d'accès aux systèmes de gestion de bases de données SGBD), environnements (systèmes d'exploitation, matériels) et principes de sécurité pour les atteindre (mécanismes réseaux, authentifications) entrent en jeu pour les acquérir.

3.2. Outils d'extraction, transformation et chargement

Plus connus sous le terme anglo-saxon Extract Transform Load (ETL), ces outils sont fondamentaux pour la construction des entrepôts de données. Ils extraient les données des systèmes hétérogènes sources, les normalisent et les rendent cohérentes entre elles, pour qu'elles puissent être utilisées ensemble. Les données sont fournies dans un format permettant leur stockage immédiat dans les entrepôts, et ultérieurement exploitables, sans recalculs par les décideurs et les analystes.

En accord avec le résultat à obtenir, et une fois les données importantes localisées dans les systèmes sources, l'outil doit les extraire, selon une fréquence déterminée (planification).
Elles sont alors stockées temporairement (staging). Cette étape et le type de fichier choisi pour ce stockage (fichiers plats, XML, tables relationnelles, etc.) sont décisifs car ils permettent de filtrer et fédérer les données afin de les rendre homogènes :

- Le filtrage sert à identifier les données aberrantes ou problématiques, notamment les données manquantes ;
- Le dédoublonnage est nécessaire lorsque plusieurs sources de données partagent des données communes ;
- Le formatage est crucial, notamment dans le cas de données codifiées (par exemple, des abréviations difficilement convertibles), ou de dates qui doivent être décomposées en un ensemble de champs (année, mois, jour, heure, minute, etc.), contenant chacun une information pertinente ;
- La dénormalisation est inévitable si la source est une base de données relationnelle, qui utilise généralement la troisième forme normale (3FN), interdisant toute redondance. À noter que le formatage et la dénormalisation peuvent être contradictoires car dans le cas de fichiers sources dont les informations sont déjà dénormalisées, il est alors préférable de les normaliser à nouveau ;
- La synchronisation garantit la cohérence des agrégats de l'entrepôt ;
- L'agrégation est une collection d'opérations possibles à effectuer sur les données. Les plus courantes sont la somme, la moyenne, le comptage, la somme cumulée, le minimum, le maximum. Ces opérations sont à considérer compte tenu du niveau de granularité de l'entrepôt.

Ces tâches conditionnent la qualité des données du système décisionnel. À ce titre, cette étape apparaît comme « la plus importante et la plus complexe à effectuer lors de l'implantation d'un entrepôt de données ».

3.3. Entrepôt de données

L'entrepôt de données « est une base de données architecturée pour des requêtes et des analyses, plutôt que pour le traitement transactionnel des données », et les résultats de ces requêtes doivent être obtenus rapidement.
L'entrepôt est organisé sur le modèle multidimensionnel évoqué précédemment. Il y a néanmoins deux types de stockage :

- L'entrepôt (data warehouse), qui concentre toutes les données ;
- Le marché de données (data mart) focalise sur une partie du métier, comme les relations clients, par exemple.

Yvan Bédard a précisé que « l'entrepôt [...] est prévu pour l'entreprise dans son ensemble alors que le marché de données est sectoriel (il peut être un sous-ensemble exact ou modifié de l'entrepôt de données) ».

3.4. Traitement analytique en ligne OLAP

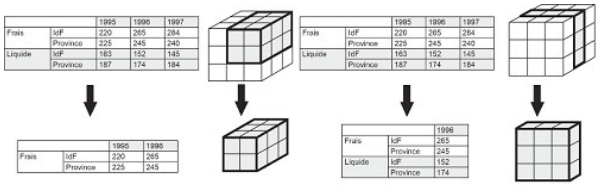
En 1993, Edgar Frank Codd introduit le terme On-Line Analytical Processing (OLAP) qui « désigne une catégorie d'applications et de technologies permettant de collecter, stocker, traiter et restituer des données multidimensionnelles à des fins d'analyses ».
Il a aussi introduit 12 « règles de base » permettant de qualifier l'OLAP :

1. Transparence : l'utilisateur doit pouvoir accéder à la base, sans se préoccuper de l'emplacement du serveur ;
2. Accessibilité : les données doivent toutes être accessibles, sans ambiguïté ;
3. Manipulation des données : la navigation doit pouvoir s'effectuer intuitivement via des interfaces ergonomiques ;
4. Souplesse d'affichage et flexibilité : le serveur doit permettre souplesse pour l'édition et réutilisation des rapports générés ;
5. Multidimensionnalité : il s'agit de la nature même d'OLAP ;
6. Client-serveur : architecture du système ;
7. Multi-utilisateur : l'accès et les recherches simultanés de la base doivent être possibles ;
8. Stabilité : les performances sont indépendantes du nombre de dimensions, ce nombre et le niveau d'agrégation doivent pouvoir être modifiés sans impact sur les temps de réponse ;
9. Gestion complète : le serveur supporte la représentation d'informations manquantes ;
10. Croisement des dimensions : le système permet d'effectuer des opérations entre et dans les dimensions ;
11. Dimensionnalité générique : toutes les dimensions d'un hypercube doivent être accessibles de manière générique, elles sont, de plus, indépendantes ;
12. Analyse sans limite : le nombre de dimensions et de niveaux d'agrégation permet des analyses complexes.

Entre entrepôt et OLAP, il n'y a qu'un pas. En effet, l'entrepôt est le lieu de stockage physique des données, tandis que l'OLAP est l'outil permettant leur analyse multidimensionnelle.

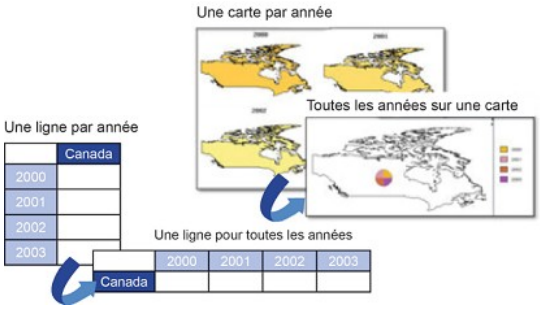
Celle-ci est l'objet d'une requête particulière, émise par l'utilisateur, *a contrario* du forage (data mining) qui vise la recherche de corrélations entre les données dans l'intégralité de l'entrepôt.
Afin de rendre l'analyse la moins contraignante et la plus souple possible, l'OLAP propose des opérateurs. Il s'agit de mécanismes servant à naviguer dans les hiérarchies et les dimensions. Les opérateurs permettent de :

- Tailler (slicing, scoping) : autorise l'extraction d'une tranche, d'un bloc d'informations. Il s'agit d'une sélection classique ;



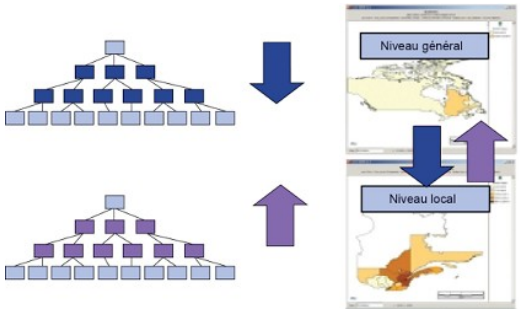
Opérateurs Slicing et Scoping

- Pivoter (rotate ou swap) : permet d'interchanger deux dimensions ;



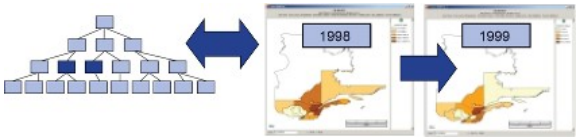
Opérateur Rotate

- Remonter (roll-up) : synthétise les informations en fonction d'une dimension. Par exemple, sur la dimension géographique, il s'agirait de passer du niveau département au niveau région ;
- Forer (drill-down) : il s'agit de l'inverse du (drill-up), on « zoome » sur une des dimensions (de la région au département) ;



Opérateurs Roll-Up et Drill-Down

- Forer latéralement (drill-accross) : en restant au même niveau de dimension, permet de changer l'une des valeurs. Par exemple, passer de l'année 1998 à l'année 1999. Le forage latéral sur une dimension spatiale peut paraître aussi simple, si l'on considère que l'on passe, par exemple, d'un département à un autre. On peut s'interroger sur la pertinence de passer de l'Ain à l'Aine. Ne serait-il pas plus pertinent de rester dans la région ? Ou de considérer des critères de voisinage ?



Opérateur Drill Across

- Percer (drill-through) : permet d'accéder au détail des informations, lorsqu'on ne dispose que de données agrégées (possible uniquement avec Hybrid OLAP).

L'architecture d'un système OLAP peut se décliner sous plusieurs formes, selon la technologie utilisée. On peut rencontrer des approches sans serveur OLAP, il s'agit alors de bases de données relationnelles, où rien n'est nativement prévu pour l'informatique décisionnelle. Il faut alors que la requête, construite dans le langage SQL (Structured Query Language), fasse état des agrégations. Ceci demande des compétences spécifiques, que tous les analystes n'ont pas forcément. L'approche ROLAP (Relationnal OLAP) est aussi basée sur une BDR, mais simulant une structure multidimensionnelle.

L'approche MOLAP (Multidimensional OLAP) est optimisée, comme son nom l'indique, pour l'analyse multidimensionnelle dont elle en gère la structure de manière physique.

HOLAP (Hybrid OLAP) est un croisement des approches MOLAP et ROLAP. Les données détaillées sont stockées dans une BDR tandis que celles agrégées le sont dans une BDM.

3.5. Outils de visualisation

Les outils de restitution sont la partie visible offerte aux utilisateurs. Par leur biais, les analystes sont à même de manipuler les données contenues dans les entrepôts et les marchés de données. Les intérêts de ces outils sont l'édition de rapports et la facilité de manipulation. En effet, la structure entière du système décisionnel est pensée pour fournir les résultats aux requêtes des utilisateurs, dans un temps acceptable (de l'ordre de quelques secondes), et sans connaissance particulière dans le domaine de l'informatique. Généralement, les outils offrent des facilités de manipulation, comme le « glisser-déposer », permettant une prise en main rapide, intuitive et conviviale.

3.6. Métadonnées

Les métadonnées, présentes à tous les niveaux, permettent de connaître les données, qu'elles soient brutes ou transformées. Moriarty et Greenwood ont déclaré, en 1997, que « les métadonnées sont aussi essentielles aux usagers que ne le sont les données elles-mêmes ». Elles décrivent le schéma de l'entrepôt, ainsi que l'ensemble des règles, des définitions, des transformations et des processus qui sont appliqués à chacune des données. Il y a deux types de métadonnées :

- Structurelles : décrivant la structure et le contenu de l'entrepôt (aussi appelées métaschéma) ;
- Accessibilité : permettant le lien entre l'entrepôt et les utilisateurs (description des données).

4. Sources

Il se peut que certaines sources aient disparu depuis leur consultation. J'ai sauvegardé une copie de la plupart d'entre elles, si besoin, je peux les produire, sous réserve de l'accord de leurs propriétaires respectifs.

- Gilles Lebrun, Christopher Charrier. 2008. *Informatique Décisionnelle*. Cours DEST CNAM UE NFE115.
- Andreas Meier. 2006. *Introduction pratique aux bases de données relationnelles. Seconde édition*. (pages 197 à 203).
- Yvan Bédard, François Létourneau, Bernard Moulin. 1998. *Perspectives d'utilisation du concept d'entrepôt de données pour les géorépertoires sur Internet (Lien 117)*.
- Claire Noirault. Novembre 2006. *Business Intelligence avec Oracle 10g : ETL, Data warehouse, Data mining, rapports…* (Lien 118)
- Thérèse Rougé-Libourel. 2008. *Entrepôts de données spatiales OLAP (Lien 119) et SOLAP.*
- Bernard Lupin. Septembre 2007. *Osez OLAP (Lien 120) – Les bases de données OLAP par l'exemple*.
- Benoît Le Rubrus. Juin 2009. *Capacité de rendu cartographique autour des technologies SOLAP (Lien 121)*. Épreuve TEST CNAM UE ENG111.
- Didier Donsez. Janvier 2006. *Principes et architectures des entrepôts de données (Lien 122)*.
- Adrien Gabriel, Elias Ohayon. 2008. *Les outils décisionnels: description de l'offre commerciale et Open Source (Lien 123)*.
- Yvan Bédard. *OLAP (Lien 124) et SOLAP : notions avancées des bases de données SIG*.
- Eric de Malleray. 24 février 2008. *Métadonnées et analyses multidimensionnelles à travers les hypercubes (Lien 125)*. Mémoire École Nationale Supérieure des Mines de Nancy – LORIA
- Frédérique Peguiron, Odile Thierry. 2005. *À propos d'un entrepôt de données universitaire : modélisation des acteurs et métadonnées (Lien 126)*.

Retrouvez l'article de Michael Tranchant en ligne : Lien 127

QlikView : principe de la base vectorielle

La puissance de QlikView tient dans sa base vectorielle, technique consistant à séparer les tables et les données. Toutefois, ceci a des conséquences sur la modélisation.

1. Un peu de vocabulaire

En matière de bases de données, on distingue les bases classiques, dites « **transactionnelles** », des bases « **décisionnelles** », telles que QlikView. Alors qu'une base transactionnelle effectue avant tout (mais pas seulement, bien sûr) des écritures et des lectures ponctuelles, une base décisionnelle est destinée à manipuler de grands volumes de données. On demande en effet à une base décisionnelle des totaux, des moyennes, des répartitions, des évolutions, etc. sur de grands ensembles de données.
Les structures de données du transactionnel ne sont pas adaptées au décisionnel. C'est la raison pour laquelle une base décisionnelle peut n'avoir qu'une lointaine ressemblance avec la base transactionnelle dont elle est issue. Il faut remodéliser.

La force d'un outil comme QlikView est d'être capable à la fois de fournir des chiffres consolidés (les ventes d'une année entière au niveau du pays, par exemple) ET de les justifier ligne à ligne. Une base classique met une éternité à produire ce genre de chose.

2. Qu'est-ce qu'une base vectorielle ?

Bien que la structure interne de QlikView soit un secret de fabrique, le principe général de la base vectorielle est connu. On extrait les données des tables (nombres, chaînes, etc.) ; on les dédoublonne et on les numérote. Les tables ne contiennent alors plus les données directement mais leur numéro.
L'avantage d'un tel système est d'éviter de stocker deux fois la même valeur. Les répétitions sont fréquentes dans les données, de sorte que le gain est en général de 90 %.

Dans l'exemple ci-dessous, la valeur « Médecin généraliste » est stockée trois fois dans la base classique contre une fois seulement dans la base vectorielle :

Base SQL

Table

Yann	DALAIN	Marseille	Médecin généraliste
Catherine	BAZIOU	La Rochelle	Rhumatologue
Jean-François	ALBERG	Lille	Médecin généraliste
Christian	SOUHAT	Lille	Ophtalmologiste
Nathalie	BAUTRET	Brest	Podologue
Floriane	MARTINO	Paris	Médecin généraliste

Base vectorielle

Table

21	7	12	14
5	2	10	19
9	1	11	14
6	20	11	16
15	3	4	18
8	13	17	14

Données

1	ALBERG
2	BAZIOU
3	BAUTRET
4	Brest
5	Catherine
6	Christian
7	DALAIN
8	Floriane
9	Jean-François
10	La Rochelle
11	Lille
12	Marseille
13	MARTINO
14	Médecin généraliste
15	Nathalie
16	Ophtalmologiste
17	Paris
18	Podologue
19	Rhumatologue
20	SOUHAT
21	Yann

Fig. 1 - Base de données classique (à gauche) et vectorielle (à droite).

3. Un deuxième avantage

Si, de plus, on a la bonne idée de trier les données (comme c'est le cas ci-dessus), comparer le numéro revient à comparer la valeur. On tient là une deuxième optimisation appréciable puisque l'ordinateur manipule les nombres beaucoup plus vite que les textes.
Bien que la structure interne de QlikView ne soit pas connue, il est probable que QlikView trie ainsi les données et ne manipule ensuite que les nombres.

4. Améliorons encore

Chaque champ possédant probablement un nombre de valeurs distinctes inférieur à celui de la base, il est tentant de numéroter les valeurs distinctes *par champ* et de garder ce numéro comme valeur de champ, étant entendu que l'on

conserve alors la table de correspondance permettant de retrouver le numéro « général » à partir du numéro « du champ ».
Avantage : des nombres plus petits dans la table, nécessitant moins de bits. Ces quelques bits gagnés ne paient pas de mine mais multipliés par le nombre de lignes et de colonnes, cela fait beaucoup !

Dans la figure ci-dessous, nous voyons que le premier champ contient des numéros de 5 à 21, ce qui nécessite cinq bits de stockage (avec cinq bits on compte jusqu'à 31). Tandis qu'avec un numéro « de champ » (à droite), le même champ ne contient plus que des numéros de 1 à 6, soit trois bits de stockage (avec trois bits on compte jusqu'à 7).

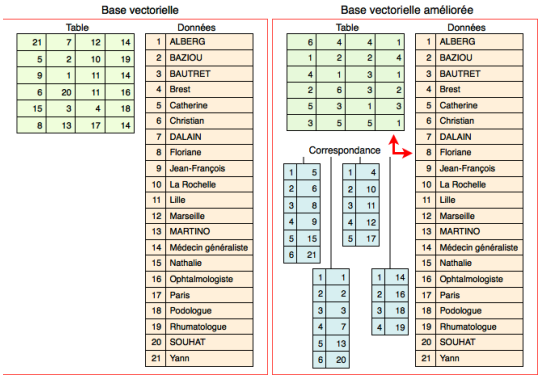


Fig. 2 - Base vectorielle standard (à gauche) et améliorée (à droite).

La structure interne de QlikView étant encore une fois tenue secrète, nous ne savons pas si la méthode utilisée est celle ci-dessus ou une autre plus performante (sûrement). Néanmoins, nous savons que le nombre de valeurs distinctes par champ est important.

5. Une première conséquence

Une première conséquence à tirer de la structure de la base vectorielle est de veiller à réduire le nombre de valeurs distinctes dans sa base. Cela signifie par exemple retirer les heures des dates, si on n'en a pas besoin, voire également le jour, toujours si on n'en a pas besoin.

6. Une deuxième conséquence

Une autre conséquence de la base vectorielle (nous n'irons pas plus loin dans cet article) concerne ce qu'on appelle les *tables de référence*, ces tables associant un code à un libellé. Dans les bases de données classiques, on a l'habitude d'en faire des sous-tables. Dans une base vectorielle, c'est à éviter !

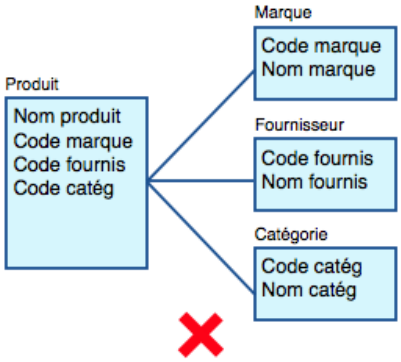


Fig. 4 - Mauvaise modélisation de tables de référence dans une base vectorielle.

Non seulement un tel schéma nécessite le stockage du code ET du libellé, le code n'étant pas forcément utile, mais ce schéma induit un travail supplémentaire de liaison entre tables.
La bonne façon de procéder (cela choque un peu au début !) est de stocker le libellé directement dans la table principale. Ceci est gagnant à tous points de vue : aucune donnée inutile n'est conservée (le code a disparu) et le schéma est plus simple :

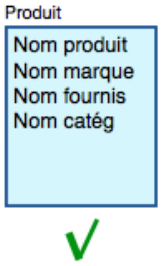


Fig. 5 - Bonne modélisation de tables de référence dans une base vectorielle.

Les données étant dédoublonnées selon le principe même de la base vectorielle, il n'y a aucune répétition.

Astuce : la fonction map ou applymap() de QlikView permet de substituer efficacement un libellé à un code. Attention, ne pas faire la substitution en SQL car cela gonflerait inutilement le trafic réseau.

De façon générale, remonter quelques champs d'une sous-table dans la table principale ne nuit pas, et simplifie le schéma. Stocker quelques attributs répétitifs ne nuit pas.

7. Conclusion

Une base vectorielle, telle que QlikView, change un certain nombre d'habitudes par rapport aux bases de données classiques (transactionnelles). Avec une base vectorielle, on cherche à réduire le nombre de lignes mais plus encore le nombre de valeurs distinctes.

P.-S. Le fichier qvd ne donne qu'une image indirecte de la structure du qvw, car la table est déconnectée des autres et ne peut plus partager ses données. Donc, malgré ce qui est dit ci-dessous, je penche plutôt pour une mise en commun globale des données dans le qvw, et non par table.

Retrouvez l'article d'Yves Ducourneau en ligne : Lien 128

Le kit du bon développeur Rails - Quelques gems à connaitre, partie 1

Tout bon développeur se doit d'avoir quelques outils pour créer sites et applications. À l'instar d'un mécanicien et de sa trousse à outils, un développeur Rails a son lot de gems indispensables.
Il existe différentes fonctionnalités que l'on va retrouver sur une très grande majorité des sites ou applications Web. Il n'est pas nécessaire avec celles-ci de tout refaire à la main à chaque fois, il existe des gems.

1. Authentification et gestion des utilisateurs

L'une des premières fonctionnalités demandées lors de la création d'un site est l'authentification d'utilisateur.

Cette fonctionnalité est particulièrement sensible en termes de sécurité, vouloir tout refaire à la main, c'est à coup sûr exposer son client à une faille.

Généralement l'utilisateur doit pouvoir créer un compte, se connecter, se déconnecter, etc. ; nous allons donc découvrir l'une des gems qui permettent de gérer tout cela.

1.1. Devise

Devise est une gem disponible pour Rails 2.3 et Rails 3. Différents modules sont disponibles suivant les besoins que vous avez. Il est possible de valider les créations de compte avec un e-mail par exemple.

Pour utiliser cette gem dans votre application, il suffit de l'ajouter à votre Gemfile :

```
gem 'devise'
```

Puis, un bundle install suffit à installer Devise. Ensuite, il vous faut configurer cette gem avant de pouvoir l'utiliser. Pour cela, il vous faut lancer la commande suivante :

```
rails generate devise:install
```

Enfin, vous pouvez créer votre MODEL (que nous appelons ici USER mais vous pouvez choisir le nom que vous souhaitez) avec Devise pour pouvoir gérer des utilisateurs :

```
rails generate devise USER
```

Suite à ceci, vous allez donc avoir un model User, ainsi qu'une migration pour créer la table en base et également de nouvelles routes dans votre fichier routes.rb. Il est maintenant possible pour un utilisateur de créer un compte, de se connecter, etc.

Vous pouvez ensuite configurer le fonctionnement afin de répondre aux fonctionnalités souhaitées.

Si vous souhaitez aller plus loin avec Devise, vous pouvez consulter le Railscast de Ryan Bates : Lien 129.

Devise n'est pas la seule gem capable de vous aider en ce qui concerne la gestion d'utilisateurs. En effet, il existe également d'autres gems telles que Authlogic par exemple : Lien 130.

Il est également possible que vous ayez besoin de permettre aux utilisateurs de se connecter via différents autres sites ou outils existants (Facebook, Twitter, LDAP…). Pour vous y aider, il y a OmniAuth (Lien 131) ; vous permettant d'intégrer la connexion via les réseaux sociaux sur votre site ou application Web : Lien 132.

Maintenant que nous avons vu comment permettre à un utilisateur de se connecter sur votre site, vous pouvez avoir besoin de gérer des droits d'accès en définissant des rôles.

1.2. Cancan

Cancan est une gem permettant de gérer les autorisations et les droits d'accès à certains contenus. En effet, vous pouvez définir ce qui est disponible ou non pour l'utilisateur connecté. Cette gem fonctionne parfaitement avec Devise ou Authlogic par exemple.

Pour installer cette gem, si vous utilisez Rails 3, il vous suffit de l'ajouter dans votre Gemfile :

```
gem 'cancan'
```

Si vous utilisez Rails 2, vous devez ajouter cette ligne dans votre fichier environment.rb (après avoir installé la gem) :

```
config.gem "cancan"
```

Ensuite, il vous faut ajouter un champ role à votre model User. Puis, il faut définir une classe Ability où se trouveront les permissions des utilisateurs. Si l'on a par exemple, des utilisateurs avec un rôle admin ou author, on initialise les droits en fonction de ce rôle.

```
class Ability
  include CanCan::Ability

  def initialize(user)
    user ||= User.new
    send user.role
  end

  def admin
    can :manage, :all
```

```
  end

  def author
    can [:create, :read, :update], [Article,
Dossier]
  end
end
```

Ici, l'admin a tous les droits alors que l'author ne peut que créer, lire ou modifier des Dossiers et Articles.

Vous pouvez contrôler les autorisations depuis le controller, par exemple pour vérifier si l'utilisateur connecté peut voir l'article qu'il a sélectionné :

```
def show
  @article = Article.find(params[:id])
  authorize! :read, @article
end
```

Ou bien, vous pouvez afficher un bloc dans une vue uniquement si l'utilisateur possède les droits nécessaires.

```
<% if can? :read, @article %>
  <%= link_to "Voir l'article",
article_path(@article) %>
<% end %>
```

Vous pouvez également gérer les erreurs lorsque l'utilisateur n'a pas accès à une page par exemple. Plus de détails sont disponibles sur le Github de Cancan : Lien 133.

2. Créer des formulaires

La gestion des utilisateurs n'est pas le seul point indispensable d'un site. La création de formulaires en fait également partie. Il est nécessaire d'avoir des formulaires pour créer ou éditer différents objets, pour se connecter… Il existe donc différentes gems pour vous aider à créer vos formulaires.

2.1. Formtastic

Formtastic (Lien 134) est une gem qui vous permet, à travers un DSL, de créer des formulaires. Cette gem est compatible avec Rails 2 et 3.

Pour pouvoir utiliser Formtastic dans votre projet, vous devez l'ajouter dans votre Gemfile puis lancer un bundle install.

```
gem 'formtastic'
```

Ensuite, il vous faut installer la gem grâce à la commande suivante :

```
rails generate formtastic:install
```

À partir de là, vous pouvez utiliser cette gem dans vos vues pour créer vos formulaires.

```
<%= semantic_form_for @article do |f| %>
  <%= f.inputs do %>
    <%= f.input :title %>
    <%= f.input :content, :input_html => { :class
=> 'article_content', :rows => 5, :cols => 50 }%>
    <%= f.input :draft, :as => :radio %>
    <%= f.input :categories, :as => :select,
:collection => Categorie.find(:all) %>
  <% end %>
  <%= f.buttons do %>
    <%= f.commit_button %>
  <% end %>
<% end %>
```

Dans le cas présent, on crée un formulaire pour un article avec un champ title qui est un input de type text, un champ content qui est un textarea (pour lequel on spécifie des options), un champ draft qui est un bouton radio et enfin un champ categories qui est un select. Enfin, on retrouve le bouton de validation du formulaire. Il vous est possible de créer une multitude d'options pour mettre en forme votre formulaire selon vos souhaits.

En plus de vous simplifier la tâche, Formtastic génère un code sémantiquement riche. Il ajoute également des balises HTML5 telles que phone, email… On y retrouve également une gestion des messages d'erreurs.

Formtastic n'est pas la seule gem qui permet une création simplifiée des formulaires. En effet il existe aussi simple_form (Lien 135) par exemple. Encore une fois, vous pouvez choisir celle qui vous convient le mieux par rapport à vos besoins et vos habitudes.

3. Moteur de recherche

Sur un site ayant un contenu suffisamment important, il peut être nécessaire d'avoir un moteur de recherche.

3.1. Ransack

Ransack est un outil qui vous permet de créer un moteur de recherche sur votre site. Il vous permet de gérer vos recherches selon deux types simple ou avancé (advanced). Pour l'utiliser, il vous faut tout d'abord l'ajouter à votre Gemfile :

```
gem 'ransack'
```

Ensuite, vous pouvez créer une recherche comme dans l'exemple ci-dessous :

```
def index
  @q = Article.search(params[:q])
  @articles = @q.result
end
```

On effectue une recherche sur les articles avec les paramètres transmis par un formulaire de recherche puis on retourne les articles correspondants.

Il faut également créer ce formulaire de recherche dans votre site afin de spécifier les critères de recherche.

```
<%= search_form_for @q do |f| %>
  <%= f.label :title_start %>
  <%= f.text_field :title_start %>
  <%= f.label :content_cont %>
  <%= f.text_field :content_cont %>
  <%= f.submit %>
<% end %>
```

On utilise le DSL pour savoir si le contenu d'un article contient bien l'expression donnée (avec cont ajouté au nom du champ) ou bien si le titre commence par un mot ou une expression donnée (avec start ajouté au nom du champ). Il existe de nombreux critères pour effectuer votre recherche.

Encore une fois, il n'y a pas une seule et unique gem pour faire cela. Ransack est une réécriture de MetaSearch (Lien 136) et d'autres gem existent.

3.2. Thinking Sphinx

Pour les applications nécessitant un moteur de recherche plus performant, vous pouvez utiliser le logiciel Sphinx : Lien 137. Celui-ci est un moteur de recherche indépendant de votre application Rails, il ne fait que générer un index à partir de la base de données. Cette indexation ne se fait d'ailleurs pas automatiquement, elle doit être invoquée régulièrement (par exemple via un cron). Afin de faire le lien entre votre application (en l'occurrence Active Record) et Sphinx, il existe une gem Thinking Sphinx.

Cette dernière vous propose un certain nombre d'options afin de gérer vos recherches. Vous pouvez par exemple configurer une taille minimum pour les mots ou encore effectuer une recherche sur des morceaux de mots.

Pour installer cette gem, il vous suffit de l'ajouter à votre Gemfile.

```
gem 'thinking-sphinx'
```

Ensuite, il vous faut définir les champs sur lesquels la recherche va s'effectuer pour le model que vous souhaitez.

```
class Article < ActiveRecord::Base
  define_index do
    indexes title, :sortable => true
    indexes content

    has created_at, updated_at
  end
end
```

Deux méthodes sont utilisées. La première, indexes, permet d'indexer les champs pour lesquels on pourra effectuer une recherche. La deuxième, has, permet de définir des attributs. Ils permettent de faire un tri par exemple.

Ensuite, il faut lancer l'indexation des champs grâce à la tâche rake suivante :

```
rake thinking_sphinx:index
```

Puis vous devez démarrer Thinking Sphinx :

```
rake thinking_sphinx:start
```

Enfin, vous pouvez effectuer une recherche sur vos articles :

```
Article.search "un article", :order =>
:created_at, :sort_mode => :desc
# on cherche les articles contenant dans leur
titre ou contenu l'expression "un article"
# et triés par ordre décroissant sur le critère
"created_at".
```

Il vous est donc possible d'effectuer des recherches sur un model donné. Vous pouvez obtenir beaucoup de détails sur le site de Thinking Sphinx : Lien 138.

4. Pagination

Certaines pages de votre site peuvent contenir des listes de résultats très longues et dans ce cas, il vaut mieux utiliser la pagination afin d'éviter les pages qui n'en finissent plus.

4.1. Will Paginate

Will Paginate est une gem permettant de paginer vos résultats très simplement : Lien 139.

Pour installer will_paginate dans une application Rails 3, il vous faut ajouter la gem dans votre Gemfile :

```
gem 'will_paginate', '~> 3.0'
```

Ensuite, il vous suffit de spécifier, dans votre controller, que les résultats sont paginés tout en indiquant les paramètres correspondants.

```
def index
  @articles = Article.paginate(:page =>
params[:page])
  # on indique la page sur laquelle l'utilisateur
se trouve
end
```

Il est également possible de spécifier le nombre de résultats par page.

```
@articles = Article.paginate(:page =>
params[:page], :per_page => 20))
```

Vous pouvez appliquer une pagination sur une relation Active Record.

```
@articles = Article.where(:draft =>
false).paginate(:page => params[:page], :per_page
=> 20))
```

Enfin, il faut ajouter la gestion des pages dans votre vue et pour cela il vous faut uniquement ajouter le bloc suivant :

```
<%= will_paginate @articles %>
```

Ceci aura pour effet d'ajouter dans votre vue HTML un bloc contenant des liens vers les pages suivantes et précédentes ainsi que quelques liens directs vers des pages précises. Il est possible de configurer l'affichage du sélecteur de page dans votre vue.

De nombreux paramètres sont à votre disposition pour obtenir ce qui convient le plus à vos besoins.

Afin d'effectuer une pagination sur votre site ou application vous pouvez également utiliser la gem kaminari : Lien 140.

5. Déploiement

Le déploiement d'une application ou d'un site Web n'est pas la partie la plus simple. Dans ce cas encore, l'usage d'une gem pour vous faciliter la tâche peut s'avérer utile.

5.1. Capistrano

Capistrano est un outil permettant d'exécuter des commandes sur différentes machines distantes : Lien 141.

On utilise pour cela un DSL afin de définir différentes tâches à exécuter. Celles-ci sont définies dans un fichier (capfile).

Une fois que vous avez créé votre fichier et que vous avez renseigné les différentes tâches à accomplir, il faut savoir sur quelles machines vous voulez déployer votre site ou application. Ceci se fait dans le fichier deploy.rb du répertoire config de votre projet. Vous pouvez obtenir plus de détails sur la façon d'utiliser capistrano dans notre article concernant le déploiement : Lien 142.

6. Conclusion

Il existe donc de nombreuses gems qui vont vous faciliter la tâche lors de l'ajout de fonctionnalités de base. Pour chacune d'entre elles, différentes gems peuvent être utilisées.

Il n'y a pas une gem parfaite dans chaque cas, c'est à vous de choisir celle qui vous semble la plus appropriée et qui vous convient le mieux.

Investissez le temps que vous ne passerez pas à développer à bien choisir vos outils et lire leur code. Posez-vous des questions. La gem est-elle activement maintenue ? Puis-je facilement remonter un bogue ? Y a-t-il une communauté active autour ? Est-elle bien testée ?

Cet investissement en veille, à l'affût des bons outils, vous sera largement récompensé à chaque fois que vous trouverez chaussure à votre pied.

Cet article a été publié avec l'aimable autorisation de Synbioz, l'article original (Le kit du bon développeur Rails, quelques gems à connaitre, partie 1 (Lien 143)) peut être vu sur le blog de Synbioz : Lien 144.

Retrouvez l'article de Synbioz en ligne : Lien 145

Le kit du bon développeur Rails - Quelques gems à connaitre, partie 2

Il est toujours bon pour un développeur de connaître quelques gems qui vont lui permettre de développer les fonctionnalités de base d'un site ou d'une application. Cet article est la suite du kit du bon développeur rails publié il y a quelques semaines sur le blog. Nous avions pu explorer des sujets tels que le déploiement, l'authentification, les formulaires de recherche mais d'autres thèmes restent à voir, nous allons en explorer une autre partie dans cet article.

1. Internationalisation

Un site multilangue entraîne forcément l'utilisation de **I18n (Internationalisation)** qui est présent dans Rails. En effet, afin de pouvoir traduire vos contenus il est nécessaire de n'avoir aucun texte en dur dans votre code. Cependant, il est parfois nécessaire de ne traduire qu'une seule partie de la page et dans ce cas il existe des gems pour vous faciliter la tâche.

1.1. I18n

I18n est donc utilisé dans Rails pour traduire tous vos textes. Vous pouvez par ailleurs obtenir plus d'informations sur la page I18n (Lien 146) du site de RubyOnRails. Pour cela, Rails se base sur la langue définie pour I18n. Il vous est possible de la définir via la ligne de commande suivante :

```
I18n.locale = :fr # pour définir français comme langue
```

Il est également possible de définir une langue par défaut pour votre application dans le fichier config/environment.rb ou config/application.rb selon la version de Rails.

```
config.i18n.default_locale = :fr
```

Pour ce qui est de la traduction en elle-même, il vous suffit de créer des fichiers (.yml ou .rb) par langue dans le répertoire *config/locales* de votre application. Il est également possible de prendre en compte d'autres fichiers en spécifiant où les trouver dans le fichier config/application.rb via la ligne suivante :

```
config.i18n.load_path += Dir[Rails.root.join('my', 'locales', '*.{rb,yml}')]
```

Il devient ensuite très simple d'avoir une application traduite dans plusieurs langues. On peut *via* le controller par exemple spécifier la langue :

```
class ApplicationController < ActionController::Base
  before_filter :set_locale

  def set_locale
    I18n.locale = params[:locale]
  end
end
```

Afin d'utiliser vos traductions, il faut utiliser la méthode translate de I18n qui retourne le texte dans la langue spécifiée. Par exemple, dans une vue, si vous avez défini la langue à français vous pouvez reprendre l'exemple qui suit :

```
<%= t("views.home_title") %>
```

Ce code vous affiche le texte défini, en français, pour la

clé home_title dans votre fichier config/locales/fr.yml :

```
fr:
  views:
    home_title: "Page d'accueil"
```

Ce fichier vous permet également de définir les libellés des attributs des models ou les messages d'erreurs.

```
fr:
  activerecord:
    attributes:
      category:
        title: "Titre de la catégorie"
```

Il est parfois nécessaire de ne traduire qu'une seule partie de la page ou du contenu saisi en base et I18n n'offre pas ces possibilités, il existe donc des gems capables de le faire.

1.2. Globalize3

Globalize3 est une gem permettant de traduire du contenu dynamique (attributs d'un modèle). Elle succède à la gem Globalize. Il est possible d'obtenir des informations sur cette gem sur la page Github : Lien 147.

Pour installer cette gem, il vous suffit de l'ajouter à votre Gemfile puis de lancer la commande bundle install :

```
gem 'globalize3'
```

Ensuite, il faut spécifier dans le model les champs qui seront traduits :

```
class Category < ActiveRecord::Base
  translates :title, :description
end
```

Une migration est alors nécessaire afin de créer la table où seront stockées les traductions :

```
class CreateCategories < ActiveRecord::Migration
  def self.up
    create_table :categories do |t|
      t.timestamps
    end
    Category.create_translation_table! :title
=> :string, :description => :text
  end

  def self.down
    drop_table :categories
    Category.drop_translation_table!
  end
end
```

Attention, pour les projets utilisant Rails 3.1 ou plus, il ne faut pas utiliser la méthode change mais continuer à utiliser up et down.

Pour une utilisation « basique » de votre application, c'est-à-dire en utilisant une seule langue par page, vous n'aurez rien à faire dans vos vues. En effet, les champs seront affichés dans la langue du site (I18n.locale) et vous allez donc avoir les contenus stockés en base affichés suivant la locale, ce qui est le but pour des sites multilangues où l'on veut juste avoir le contenu traduit suivant la langue choisie par l'utilisateur.

Cependant, si vous souhaitez utiliser plusieurs langues sur une même page, ce qui est demandé dans certains cas, il vous faut utiliser les blocs définis par Globalize3. Dans les vues de votre projet Rails, ces derniers servent à traduire une partie de votre contenu dans une langue choisie auparavant :

```
<%
  my_locale = :fr
  # my_locale peut également être un helper ou
une variable définie dans le controller.
%>

<% Globalize.with_locale(my_locale) do %>
  <% render :partial => "my_partial_with_locale"
%>
<% end %>
```

On peut donc avoir le contenu des pages en anglais et uniquement la partie qui se trouve incluse dans le bloc Globalize traduite en français par exemple. Il est donc très simple d'utiliser plusieurs langues dans une même vue.

2. Gestion de fichiers

La gestion de fichiers est un autre point important lors de la création d'une application ou d'un site Web. On peut avoir des images à stocker si on permet à l'utilisateur de les charger par exemple. Différentes gems existent pour vous aider dans cette démarche.

2.1. Paperclip

Paperclip est une gem qui permet de gérer des fichiers suite au chargement par un utilisateur. Il est possible d'obtenir des informations sur cette gem sur la page Github de Paperclip : Lien 147. Afin de pouvoir utiliser cette dernière, il est nécessaire d'avoir installé imagemagick. Ensuite, pour installer Paperclip, il vous suffit de l'ajouter dans votre Gemfile et lancer un bundle install :

```
gem "paperclip"
```

Dans le model, il faut spécifier le champ qui concerne le fichier à stocker :

```
class Category < ActiveRecord::Base
  has_attached_file :logo
end
```

Il est possible, si le fichier en question est une image, de créer différentes versions du fichier, par exemple, stocker l'image dans différents formats :

```
class Category < ActiveRecord::Base
  has_attached_file :logo, :styles => { :large =>
"600x300>", :medium => "200x100", :thumb =>
"100x50>" }
  # dans ce cas l'image sera stockée quatre
fois : l'image de base telle qu'elle est uploadée
  # ainsi que les trois formats définis ci-dessus
end
```

Ensuite, la migration vous permettra de créer les différents champs utilisés par Paperclip :

```
class AddLogoToCategory < ActiveRecord::Migration
  def self.up
    change_table :categories do |t|
      t.has_attached_file :logo
    end
  end

  def self.down
    drop_attached_file :categories, :logo
  end
end
```

Après avoir lancé un rake db:migrate, vos champs sont donc en base et vous pouvez donc charger un logo pour vos catégories dans votre formulaire de création ou d'édition :

```
<%= form_for @category, :html => { :multipart =>
true } do |f| %>
  <%= form.label :title %>
  <%= form.text_field :title %>

  <%= form.label :logo %>
  <%= form.file_field :logo %>
<% end %>
```

Par la suite, vous pourrez afficher les images en question dans les formats souhaités :

```
# pour afficher l'image telle qu'elle a été
uploadée
<%= image_tag @category.logo.url %>

# pour afficher l'image en format 'medium'
<%= image_tag @category.logo.url(:medium) %>
```

Pour supprimer un fichier chargé, il suffit de mettre la valeur du champ en question à nil :

```
@category.logo = nil
```

Il est possible de faire fonctionner Paperclip avec le service de stockage d'Amazon S3 via une autre gem, aws-sdk : Lien 149.

D'autres exemples d'utilisation de Paperclip sont disponibles dans un Railscast de Ryan Bates : Lien 150.

Encore une fois, Paperclip n'est pas la seule gem disponible afin de vous aider dans la gestion des fichiers.

2.2. Dragonfly

Tout comme Paperclip, Dragonfly est une gem permettant de gérer les fichiers dans votre application Rails. Des informations sont disponibles sur la page Github de Dragonfly : Lien 151.

Afin d'installer Dragonfly, il vous faut l'ajouter à votre Gemfile :

```
gem 'rack-cache', :require => 'rack/cache'
gem 'dragonfly', '~>0.9.10'
```

La gem rack-cache est nécessaire au fonctionnement de Dragonfly.

Ensuite, ajoutez la ligne suivante dans le fichier config/initializers/dragonfly.rb (qu'il vous faudra créer auparavant s'il n'existe pas) :

```
require 'dragonfly/rails/images'
```

Vous pouvez maintenant créer la migration qui permettra de stocker le fichier :

```
class AddLogoToCategory < ActiveRecord::Migration
  def self.up
    add_column :categories, :logo_uid, :string
  end

  def self.down
    remove_column :categories, :logo_uid
  end
end
```

Puis, dans le model en question (dans notre cas Category), vous devez ajouter un image_accessor :

```
class Category < ActiveRecord::Base
  image_accessor :logo
end
```

Il vous est désormais possible de déposer les logos de vos catégories via un formulaire, dans l'une des vues de votre projet :

```
<%= form_for @category, :html => { :multipart =>
true } do |f| %>
  <%= form.label :title %>
  <%= form.text_field :title %>

  <%= form.label :logo %>
  <%= form.file_field :logo %>
<% end %>
```

Afin de supprimer un fichier, vous pouvez utiliser une checkbox dans votre formulaire :

```
<%= f.check_box :remove_logo %>
```

Si cette dernière a pour valeur true, alors le fichier sera supprimé.

Pour ce qui est de l'affichage d'une image, contrairement à Paperclip, on ne définit pas les formats d'une image lors de la sauvegarde mais lors de l'affichage.

```
# pour afficher l'image telle qu'elle a été
uploadée
<%= image_tag @category.logo.url %>

# pour afficher l'image dans un autre format
<%= image_tag @category.logo.thumb("200x100").url
%>
```

Dragonfly et Paperclip sont donc deux gems pour la gestion de fichiers. Chacun peut choisir celui avec lequel il a le plus d'affinités et qu'il trouve le mieux adapté à ses

besoins.

3. Conclusion

Ces deux articles offrent un bon tour d'horizon des solutions actuelles pour les besoins récurrents. Dans la dernière partie, nous évoquerons les tests et les outils annexes.

Connaître ces gems peut s'avérer très pratique, c'est un gain de temps dans la plupart des cas, cela évite de repartir de zéro lors du développement de certaines fonctionnalités.

Au-delà des gems présentées, il existe des gems pour presque toutes les fonctionnalités, il vous est donc facile de trouver celle qui est la plus adaptée pour vous.

De plus, la plupart des gems sont disponibles sur Github et il est donc très simple de remonter un bogue ou de poser une question. En outre, ces gems sont utilisées par de très nombreux développeurs, on peut donc considérer qu'elles sont testées et approuvées par la communauté.

Il n'y a donc aucune raison pour ne pas utiliser les gems existantes plutôt que de développer soi-même des fonctionnalités de base.

Cet article a été publié avec l'aimable autorisation de Synbioz, l'article original (Le kit du bon développeur Rails, quelques gems à connaitre, partie 2 (Lien 152)) peut être vu sur le blog de Synbioz : Lien 144.

Retrouvez l'article de Synbioz en ligne : Lien 153

Personnalisez le menu contextuel avec Firefox

HTML5 redéfinit certains aspects et certaines balises du HTML.
Parmi les éléments qui retrouvent une nouvelle jeunesse figure la balise menu (Lien 154).
Bien sûr, évoquer cette balise ne présente que peu d'intérêt puisqu'elle n'est, à l'heure actuelle (et sauf erreur de ma part) reconnue par aucun navigateur...

Par aucun navigateur ? Vraiment ?

En fait, pas tout à fait : parmi les différentes utilisations de la balise *menu*, une est utilisable par Firefox depuis la version 8 : la personnalisation du menu contextuel lié à un élément de la page (ou de la page elle-même).

La syntaxe de la balise *menu* est la suivante :

```
<menu type="..." id="...">
```

où *type* correspond au comportement attendu pour le menu et *id* l'identifiant du menu.
Cette balise peut (dans le cadre du menu contextuel) contenir soit des sous-menus représentés par une nouvelle balise *menu* soit des éléments de menu représentés par des balises *menuitem*.

Bon, assez d'explications théoriques, passons à la pratique avec un exemple :

```
<menu type="context" id="developpez">
  <menuitem label="Aller sur developpez.com"
onclick="location.href='http://www.developpez.com
'"
icon="http://www.developpez.com/template/favicon.
ico"></menuitem>
  <menu label="Forums developpez.com">
    <menuitem label="(X)HTML"
onclick="location.href='http://www.developpez.net
/forums/f39/webmasters-developpement-web/html-
dhtml-xhtml/'"
icon="http://www.developpez.com/template/favicon.
ico"></menuitem>
    <menuitem label="CSS"
onclick="location.href='http://www.developpez.net
/forums/f463/webmasters-developpement-web/css/'"
icon="http://www.developpez.com/template/favicon.
ico"></menuitem>
    <menuitem label="JavaScript"
onclick="location.href='http://www.developpez.net
/forums/f23/webmasters-developpement-
web/javascript/'"
icon="http://www.developpez.com/template/favicon.
ico"></menuitem>
    <menuitem label="AJAX"
onclick="location.href='http://www.developpez.net
/forums/f458/webmasters-developpement-web/ajax/'"
icon="http://www.developpez.com/template/favicon.
ico"></menuitem>
  </menu>
</menu>
```

Dans cet exemple (relativement simple), nous ajoutons des liens vers developpez.com et différents forums du site.
Vous pouvez constater que le type de menu est *context* pour préciser qu'il s'agit de personnaliser le menu contextuel.
D'autre part, le sous-menu et les éléments des menus possèdent des attributs *label* indiquant le texte à afficher correspondant à chaque option.
Les éléments de menus possèdent quant à eux des attributs *icon* (image apposée à côté de l'élément) et *onclick* (action JavaScript à lancer au clic sur cet élément).

Pour autant, vous aurez beau ajouter cette portion de code dans votre page et faire des clics droits partout dans la page (sur Firefox bien entendu) pour tester, rien ne se passera !
En fait, nous avons oublié l'essentiel : déterminer à quel élément de la page le menu contextuel est associé.
Pour préciser cela, il suffit d'ajouter un attribut *contextmenu* à l'élément souhaité. La valeur de cet attribut doit correspondre à l'identifiant (attribut *id*) du menu.
Dans l'exemple suivant, nous posons cet attribut sur la balise *body* afin que le menu contextuel s'applique sur toute la page :

```
<!doctype html>
<html lang="fr">
<head>
  <meta charset="utf-8" />
  <title>Menu contextuel</title>
  <style>
    html, body{
      height: 100%
    }
    h1{
      text-align: center;
    }
  </style>
</head>
<body contextmenu="developpez">
  <h1>Exemple de menu contextuel
personnalisé</h1>
  <div>Faites un clic droit dans la page pour
voir les options ajoutées au menu
contextuel.</div>
<menu type="context" id="developpez">
  <menuitem label="Aller sur developpez.com"
onclick="location.href='http://www.developpez.com
'"
icon="http://www.developpez.com/template/favicon.
ico"></menuitem>
  <menu label="Forums developpez.com">
    <menuitem label="(X)HTML"
onclick="location.href='http://www.developpez.net
/forums/f39/webmasters-developpement-web/html-
dhtml-xhtml/'"
icon="http://www.developpez.com/template/favicon.
ico"></menuitem>
    <menuitem label="CSS"
onclick="location.href='http://www.developpez.net
/forums/f463/webmasters-developpement-web/css/'"
```

```
icon="http://www.developpez.com/template/favicon.ico"></menuitem>
    <menuitem label="JavaScript" onclick="location.href='http://www.developpez.net/forums/f23/webmasters-developpement-web/javascript/'" icon="http://www.developpez.com/template/favicon.ico"></menuitem>
    <menuitem label="AJAX" onclick="location.href='http://www.developpez.net/forums/f458/webmasters-developpement-web/ajax/'" icon="http://www.developpez.com/template/favicon.ico"></menuitem>
  </menu>
</menu>
</body>
</html>
```

Notre exemple est maintenant fonctionnel. Vous pouvez voir un exemple en ligne : Lien 155.

Retrouvez ce billet blog de Bovino en ligne : Lien 156

Les derniers tutoriels et articles

Débuter avec Node.js partie 3 - Première application Node.js et HTML5

Suite de la série de billets consacrés à Node.js. Après une petite introduction de ce serveur JavaScript puis une présentation de Cloud9 IDE, nous allons enfin mettre les mains dedans !

1. Introduction

Pour ce *Quick Start*, je vous propose une application qui fera du *push* sur tous les clients pour leur envoyer l'heure système ainsi que le nombre de clients actuellement connectés au serveur (avec gestion de la déconnexion). Ces clients pourront envoyer des *pokes* aux autres clients, le tout tournera avec express (Lien 157) et Socket.IO (Lien 158).

L'idée n'est pas de développer une application de gestion complète (qui mériterait une bonne dizaine de billets) ni de faire un tour d'horizon de la centaine de bibliothèques existantes, mais juste de vous montrer à quoi ressemble le code d'une application Node avec deux bibliothèques très répandues dans la communauté.

2. Installation

Normalement, vous devriez avoir Node et NPM installés sur votre machine (Débuter avec Node.js partie 1 § installation : Lien 159). Pour rappel, selon votre OS et votre *package manager* si besoin (ici Mac OSX et Homebrew) :

```
rmat0n:~$ brew install node
rmat0n:~$ curl http://npmjs.org/install.sh | sudo sh
```

Dès lors, vous n'avez plus que deux commandes à effectuer pour installer les modules express et Socket.IO pour Node :

```
rmat0n:~$ npm install express
rmat0n:~$ npm install socket.io
```

Vous voilà prêt, commençons tout de suite !

3. Arborescence

Le projet aura l'arborescence suivante :

```
/ project
    / public
        index.html
    server.js
```

Très simple donc avec un fichier server.js et dans un dossier public, un fichier index.html.

4. Client

Tout d'abord, voyons le squelette du fichier HTML :

```
<html>
<head>
  <title>Socket IO Demo</title>
  <script src="http://code.jquery.com/jquery-1.5.min.js"></script>
  <script src="/socket.io/socket.io.js"></script>
</head>
<body>
  <div>
    <h2>Démo Node.js + Socket IO</h2>
    <label for="timestamp">Timestamp</label>
    <div id="timestamp"></div>
    <label for="clients">Clients</label>
    <div id="clients"></div>
    <label for="message">Message</label>
    <div id="message"></div>
    <p><button id="poke">Send poke !</button></p>
  </div>
</body>
</html>
```

Un fichier très standard donc, avec un import du dernier jQuery, l'import de Socket.IO (module chargé par Node) et quelques div qui me permettront d'afficher les retours du serveur.

À la suite des deux balises script déjà présentes, je rajoute le code JavaScript suivant :

```
<script type="text/javascript">
    $(document).ready(function () {
      var sock = new io.Socket();
      sock.on('message', function (data) {
        var obj = JSON.parse(data);
        if(obj.message) {
          $('#message').text(obj.message);
        } else {
          $('#timestamp').text(obj.timestamp);
          $('#clients').text(obj.clients);
        }
      });
```

```
      sock.connect();
      $("#poke").click(function()
{ sock.send("Poke !"); });
    });
  </script>
```

Quelques explications :

- tout d'abord, je crée un object io.Socket (qui par défaut utilisera le host d'entrée et le port 80) ;
- lorsque le serveur m'envoie un message, je peux le récupérer côté client en utilisant le code suivant (API Socket.IO) : sock.on('message', function(data) {...}); ;
- le serveur me renverra un objet JSON, je le parcours et récupère les propriétés message ou timestamp et client que je mets dans les champs associés ;
- je connecte la socket immédiatement après ;
- je définis un gestionnaire d'événement sur le bouton et lors d'un clic utilisateur, j'envoie un message poke.

Plutôt simple non ? Même si l'API standard est déjà très simple d'utilisation, Socket.IO nous simplifie encore plus l'utilisation de *Web Sockets* côté client et côté serveur.

Voyons maintenant le code côté serveur.

5. Serveur

Et voici le fichier server.js :

```
var io = require('socket.io');
var express = require('express');

var app = express.createServer();
app.configure(function(){
  app.use(express.static(__dirname + '/public'));
});
app.get('/', function(req, res, next){
  res.render('./public/index.html');
});
app.listen(8333);

var socket = io.listen(app, {
  flashPolicyServer: false,
  transports: ['websocket', 'flashsocket',
'htmlfile', 'xhr-multipart', 'xhr-polling',
'jsonp-polling']
});
var allClients = 0;
var clientId = 1;
socket.on('connection', function(client) {
  var my_timer;
  var my_client = { "id": clientId, "obj": client
};
  clientId += 1;
  allClients += 1;
  my_timer = setInterval(function () {
    my_client.obj.send(JSON.stringify({ "timestam
p": (new Date()).getTime(), "clients": allClients
}));
  }, 1000);
  client.on('message', function(data) {
    my_client.obj.broadcast(JSON.stringify({ mess
age: "poke send by client "+my_client.id }));
    console.log(data);
  });
  client.on('disconnect', function() {
    clearTimeout(my_timer);
    allClients -= 1;
    console.log('disconnect');
  });
});
```

Dans le détail :

- j'importe les modules Socket.IO et express ;
- je crée le serveur express, je configure le dossier public en statique, je *mappe* l'URL racine '/' vers le fichier index.html et je démarre le serveur ; on voit ici que express nous simplifie grandement le mapping entre une URL et une action (mais il n'est pas le seul) avec une bonne gestion des paramètres, j'aime beaucoup cette bibliothèque car elle me rappelle le très puissant Spring MVC : Lien 160;
- je crée ensuite l'objet socket en spécifiant quelques transports et en supprimant le *flash policy server* (inutile pour notre exemple) ;
- lorsqu'un client se connecte, je le stocke dans un objet JSON, j'augmente la variable id et j'augmente le nombre de clients connectés (variable allClients) ;
- je lance un setInterval pour envoyer toutes les secondes l'heure courante du serveur ainsi que le nombre de clients connectés (le JSON qui sera récupéré dans mon client, *cf.* code du § client) ;
- si un client envoie un message côté serveur, je le répercute à tous les autres avec un message générique mentionnant l'id du client qui a envoyé le message ;
- enfin, sur une déconnexion d'un client, je stoppe le setInterval lié à ce client et je décrémente la variable allClients ; celle-ci sera ensuite envoyée à tous les autres clients lors d'un prochain envoi de données du setInterval.

Il y a bien sûr différentes manières de faire ce code, celui-ci ne sert qu'à vous montrer ce que l'on peut faire :

- envoyer un message à un client spécifique ;
- renvoyer un message d'un client vers tous les autres ;
- mettre à jour des variables globales utilisées par tous les clients ;
- …

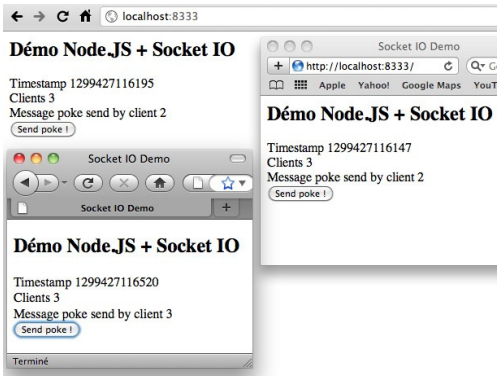
D'ailleurs, si vous avez d'autres astuces concernant les *Web Sockets* dans Node, n'hésitez pas à les partager en commentaires !

6. Run baby !

Bon, il n'y a plus qu'à lancer le code et à vous rendre à l'URL http://localhost:8333/ :

```
rmat0n:~/dev/node-example$ node server.js
```

Et maintenant, amusez-vous à lancer plusieurs navigateurs à cette URL, à envoyer des pokes, à vous déconnecter…



7. Conclusion

Bien sûr, on est très loin d'une application de gestion et ce n'est pas le but de cet article. Mais vous pouvez voir avec cet exemple un code très minimal qui fait des Web Sockets, du send, du broadcast, du MVC… le tout très facilement !

Prochaine étape et dernier article sur Node : déployer ce projet sur un serveur Node et pour cela nous allons regarder de plus près l'offre de Nodester (Lien 161).

Pour ceux qui veulent récupérer le projet, le code se trouve ici : Lien 162.

Cet article a été publié avec l'aimable autorisation de Web Tambouille, l'article original (Node.JS partie 3 - Première application Node.js et HTML5 : Lien 163) peut être vu sur Web Tambouille : Lien 164.

Retrouvez l'article de Romain Maton en ligne : Lien 165

Liens

Lien 01 : http://cpp.developpez.com/actu/37859/Un-nouveau-C-est-ne-la-norme-ISO-C-11-finale-vient-d-etre-publiee/
Lien 02 : http://qt.developpez.com/actu/38218/Le-Qt-Project-est-la-le-projet-d-open-gouvernance-pour-le-framework-C-est-arrive-a-terme/
Lien 03 : http://qt-project.org/wiki/Qt_Creator_Releases
Lien 04 : http://www.developpez.net/forums/d1199070/c-cpp/bibliotheques/qt/edi/qt-creator/qt-creator-2-5-sorti/#post6678234
Lien 05 : http://www.developpez.net/forums/d1199070/c-cpp/bibliotheques/qt/edi/qt-creator/qt-creator-2-5-beta-sorti/
Lien 06 : http://qt.developpez.com/actu/38218/Le-Qt-Project-est-la-le-projet-d-open-gouvernance-pour-le-framework-C-est-arrive-a-terme/
Lien 07 : http://lists.qt-project.org/mailman/listinfo/pyside
Lien 08 : https://bugreports.qt-project.org/
Lien 09 : http://wiki.qt-project.org/The_Qt_Governance_Model
Lien 10 : http://www.developpez.net/forums/d1198492/autres-langages/python-zope/gui/pyside-pyqt/pyside-devient-add-on-qt/
Lien 11 : http://www.developpez.net/forums/d1198305/c-cpp/bibliotheques/qt/presentation-nouvelle-ere-dihm-automobiles-interface-utilisateur-3d/
Lien 12 : http://labs.qt.nokia.com/2011/05/09/thoughts-about-qt-5/
Lien 13 : http://labs.qt.nokia.com/2011/05/11/responses-to-qt-5/
Lien 14 : http://qt-project.org/wiki/Qt-Platform-Abstraction
Lien 15 : https://launchpad.net/%7Eforumnokia/+archive/fn-ppa
Lien 16 : http://qt-project.org/wiki/Building_Qt_5_from_Git
Lien 17 : http://releases.qt-project.org/qt5.0/alpha/
Lien 18 : http://qt-project.org/wiki/Qt-5-Alpha-building-instructions
Lien 19 : http://www.developpez.net/forums/d1209764/c-cpp/bibliotheques/qt/sortie-qt-5-alpha/
Lien 20 : http://qt.developpez.com/actu/41531/Deploiement-d-applications-Qt-Commercial-sur-les-tablettes-Windows-8-seul-un-changement-de-style-est-necessaire/
Lien 21 : http://www.digia.com/en/Blogs/Qt-blog/Sami-Makkonen/Dates/2012/1/2012/
Lien 22 : http://msdn.microsoft.com/en-us/library/hh202906%28v=VS.92%29.aspx
Lien 23 : http://msdn.microsoft.com/en-us/library/windows/desktop/aa511282.aspx
Lien 24 : http://qt.developpez.com/doc/4.7/qml-gridview.html
Lien 25 : http://www.digia.com/en/blogs/Qt-blog/?feed=rss
Lien 26 : http://qt.developpez.com/doc/4.7/qml-xmllistmodel.html
Lien 27 : http://qt.developpez.com/doc/4.7/qgraphicsview.html
Lien 28 : http://qt.developpez.com/doc/4.7/qpropertyanimation.html
Lien 29 : http://qt.developpez.com/doc/4.7/qeasingcurve.html#Type-enum
Lien 30 : http://www.youtube.com/v/cTEMkDZxpaw
Lien 31 : ftp://ftp-developpez.com/qt-digia/tutoriels/qt/applications-style-metro/fichiers/QtMetroStyle.zip
Lien 32 : http://qt-digia.developpez.com/tutoriels/qt/applications-style-metro/
Lien 33 : http://www.boost.org/doc/libs/1_48_0/libs/graph/doc/using_adjacency_list.html#sec:choosing-graph-type
Lien 34 : http://matthieu-brucher.developpez.com/tutoriels/cpp/boost/graph/implementation/
Lien 35 : http://www.boost.org/doc/libs/1_48_0/libs/graph/doc/index.html
Lien 36 : http://gbelz.developpez.com/boost/graph/
Lien 37 : http://adiguba.developpez.com/tutoriels/java/7/
Lien 38 : http://www.touilleur-express.fr/2009/07/27/senior
Lien 39 : http://hakanai.free.fr/
Lien 40 : http://www.leszindeps.fr/
Lien 41 : http://www.localizeyourapps.com/
Lien 42 : http://www.lateral-thoughts.com/
Lien 43 : http://www.devoxx.com/display/FR12/Speakers
Lien 44 : https://cfp.devoxx.com/
Lien 45 : http://lescastcodeurs.com/2012/03/les-cast-codeurs-podcast-episode-55-la-tva-est-en-sus/
Lien 46 : http://www.devoxx.com/display/FR12/Devoxxians
Lien 47 : http://thierry-leriche-dessirier.developpez.com/articles/interview/hugo-lassiege-devoxx-france/
Lien 48 : http://alain-bernard.developpez.com/tutoriels/eclipse/intro-zest
Lien 49 : http://eclipse.developpez.com/cours/?page=platform-cat#plugin-dev
Lien 50 : http://eclipse.developpez.com/cours/?page=platform-cat#ihm-dev
Lien 51 : ftp://ftp-developpez.com/alain-bernard/tutoriels/eclipse/intro-zest/src-intro-zest.zip
Lien 52 : http://alain-bernard.ftp-developpez.com/tutoriels/eclipse/intro-zest/src-intro-zest.zip
Lien 53 : http://www.eclipse.org/pde/incubator/dependency-visualization/index.php
Lien 54 : http://eclipse.developpez.com/cours/?page=platform-cat
Lien 55 : http://www.eclipse.org/gef/zest/
Lien 56 : http://wiki.eclipse.org/index.php/GEF_Zest_Visualization
Lien 57 : http://www.eclipse.org/gef/zest/snippets.php
Lien 58 : http://alain-bernard.developpez.com/tutoriels/eclipse/complements-zest/
Lien 59 : http://www.developpez.net/forums/d1191152/java/general-java/java-mobiles/android/android-850-000-activations-jour/
Lien 60 : http://www.developpez.net/forums/d1199335/systemes/autres-systemes/mobiles/l-interet-developpeurs-android-baisse-faveur-html-5-a/
Lien 61 : http://www.developpez.com/actu/38650
Lien 62 : http://www.developpez.net/forums/d1188998/systemes/linux/distributions/ubuntu/ubuntu-for-android-nouveau-projet-canonical/
Lien 63 : http://www.youtube.com/watch?v=GdZxbmEHW7M
Lien 64 : http://www.developpez.net/forums/u412164-a367-i1219.png
Lien 65 : https://play.google.com/
Lien 66 : http://www.developpez.net/forums/u412164-a367-i1218.png
Lien 67 : http://www.developpez.net/forums/d1194110/club-professionnels-informatique/actualites/google-play-google-centralise-services-multimedia/
Lien 68 : http://msdn.microsoft.com/fr-fr/library/8f1hz171.aspx
Lien 69 : http://rapidapplicationdevelopment.blogspot.com/2007/01/parameter-passing-in-c.html
Lien 70 : http://www.yoda.arachsys.com/csharp/parameters.html
Lien 71 : http://www.yoda.arachsys.com/csharp/memory.html
Lien 72 : http://tlevesque.developpez.com/tutoriels/csharp/passage-parametres-csharp/
Lien 73 : http://csharpindepth.com/Articles/Chapter8/PropertiesMatter.aspx
Lien 74 : http://blogs.msdn.com/ricom/archive/2006/09/07/745085.aspx
Lien 75 : http://tlevesque.developpez.com/tutoriels/csharp/pourquoi-les-proprietes-sont-importantes/

Lien 76 : http://jormes.developpez.com/traductions/casse-tetes-csharp-reponses/
Lien 77 : http://www.yoda.arachsys.com/csharp/teasers.html
Lien 78 : http://www.ayende.com/Blog/archive/2007/12/31/Tricky-Code.aspx
Lien 79 : http://jormes.developpez.com/traductions/casse-tetes-csharp/
Lien 80 : http://www.developpez.net/forums/d1197571/webmasters-developpement-web/general-conception-web/webmarketing/google-bientot-capable-repondre-aux-questions/
Lien 81 : http://www.w3.org/TR/sparql11-query/
Lien 82 : http://jplu.developpez.com/tutoriels/web-semantique/nouveautes-sparql-1-1/
Lien 83 : http://www.w3.org/TR/rdf11-concepts/
Lien 84 : http://www.w3.org/TR/turtle/
Lien 85 : http://json-ld.org/spec/latest/json-ld-syntax/
Lien 86 : http://jplu.developpez.com/tutoriels/web-semantique/nouveautes-rdf-1-1/
Lien 87 : http://www.w3.org/TR/2011/WD-rdfa-core-20110331/
Lien 88 : http://ivan-herman.name/2010/04/22/rdfa-1-1-drafts/
Lien 89 : http://www.w3.org/QA/2011/05/rdfa_11_with_a_rich_snippet_ex.html
Lien 90 : http://tcuvelier.developpez.com/tutoriels/web-semantique/rdfa/introduction/
Lien 91 : http://tcuvelier.developpez.com/tutoriels/web-semantique/rdfa/nouveautes-rdfa-1-1/
Lien 92 : http://djibril.developpez.com/tutoriels/perl/installation-modules/
Lien 93 : http://www.zlib.net/
Lien 94 : http://djibril.developpez.com/tutoriels/perl/comment-installer-bibliotheque-gd/fichiers/zlib-1.2.6.tar.gz
Lien 95 : http://libjpeg.sourceforge.net/
Lien 96 : http://djibril.developpez.com/tutoriels/perl/comment-installer-bibliotheque-gd/fichiers/jpegsrc.v6b.tar.gz
Lien 97 : http://sourceforge.net/projects/libpng/
Lien 98 : http://djibril.developpez.com/tutoriels/perl/comment-installer-bibliotheque-gd/fichiers/libpng-1.2.48.tar.gz
Lien 99 : http://www.freetype.org/index2.html
Lien 100 : http://djibril.developpez.com/tutoriels/perl/comment-installer-bibliotheque-gd/fichiers/freetype-2.4.9.tar.gz
Lien 101 : http://www.libgd.org/
Lien 102 : http://djibril.developpez.com/tutoriels/perl/comment-installer-bibliotheque-gd/fichiers/pierrejoye-gd-libgd-5551f61978e3.tar.gz
Lien 103 : http://djibril.developpez.com/tutoriels/perl/comment-installer-bibliotheque-gd/
Lien 104 : http://support.microsoft.com/kb/176670
Lien 105 : http://notepad-plus-plus.org/fr/
Lien 106 : http://warin.developpez.com/access/fichiers/
Lien 107 : http://arkham46.developpez.com/articles/office/vba64bits/
Lien 108 : http://arkham46.developpez.com/articles/access/utilisateurs-ldb/#modules
Lien 109 : http://msdn.microsoft.com/en-us/library/windows/desktop/aa364232%28v=vs.85%29.aspx
Lien 110 : http://arkham46.developpez.com/articles/access/utilisateurs-ldb/#jetlock
Lien 111 : http://msdn.microsoft.com/en-us/library/aa164890%28v=office.10%29.aspx
Lien 112 : ftp://ftp-developpez.com/arkham46/articles/access/utilisateurs-ldb/fichiers/ldbviewer.zip
Lien 113 : http://msdn.microsoft.com/en-us/library/aa155436%28v=office.10%29.aspx
Lien 114 : http://argyronet.developpez.com/downloads/tools/mdbviewer/v4/
Lien 115 : http://arkham46.developpez.com/articles/access/utilisateurs-ldb/
Lien 116 : http://tranchant.name/2011/11/informatique-decisionnelle/
Lien 117 : http://sirs.scg.ulaval.ca/Yvanbedard/article_nonprotege/224.pdf
Lien 118 : http://books.google.fr/books?id=2StX-h64ShgC&lpg=PT9&dq=Entrepôts%20de%20données&pg=PT9
Lien 119 : http://www.lirmm.fr/%7Elibourel/FMIN206/cours11_BDS-OlapSolap.pdf
Lien 120 : http://bernard.lupin.pagesperso-orange.fr/
Lien 121 : http://blerubrus.free.fr/cnam/ueeng111/solap.pdf
Lien 122 : http://membres-liglab.imag.fr/donsez/cours/bddwdm.pdf
Lien 123 : http://webloria.loria.fr/%7Epeguiron/miage/Projet_Bibliographique.pdf
Lien 124 : http://www.scribd.com/doc/6964842/OLAP-et-SOLAP-complet-avec-explication-ppt-univ-laval
Lien 125 : http://www.loria.fr/%7Epeguiron/mines/Projet_recherche_FICM3_Eric-de%20Malleray.pdf
Lien 126 : http://hal.archives-ouvertes.fr/docs/00/03/69/39/PDF/eda2005.pdf
Lien 127 : http://mtranchant.developpez.com/tutoriels/Business-Intelligence/qu-est-que-informatique-decisionnelle/
Lien 128 http://yves-ducourneau.developpez.com/tutoriels/Business-Intelligence/qlikview-principe-base-vectorielle/
Lien 129 http://railscasts.com/episodes/209-introducing-devise
Lien 130 : https://github.com/binarylogic/authlogic
Lien 131 : https://github.com/intridea/omniauth
Lien 132 : http://www.synbioz.com/blog/2011/10/18/integrer_les_reseaux_sociaux_dans_son_application_rails_avec_oauth
Lien 133 : https://github.com/ryanb/cancan
Lien 134 : https://github.com/justinfrench/formtastic
Lien 135 : https://github.com/plataformatec/simple_form
Lien 136 https://github.com/ernie/meta_search
Lien 137 : http://sphinxsearch.com/
Lien 138 : http://freelancing-god.github.com/ts/fr/
Lien 139 : https://github.com/mislav/will_paginate
Lien 140 : https://github.com/amatsuda/kaminari
Lien 141 : https://github.com/capistrano/capistrano
Lien 142 : http://www.synbioz.com/blog/2011/11/15/deployer_et_heberger_une_application_rails_avec_capistrano_et_unicorn
Lien 143 : http://www.synbioz.com/blog/2011/12/28/le_kit_du_bon_developpeur_rails_partie_1
Lien 144: http://www.synbioz.com/blog/
Lien 145 : http://synbioz.developpez.com/tutoriels/ruby/kit-developpeur-rail-1/
Lien 146 : http://guides.rubyonrails.org/i18n.html
Lien 147 : https://github.com/svenfuchs/globalize3
Lien 148 : https://github.com/thoughtbot/paperclip
Lien 149 http://rubydoc.info/gems/paperclip/Paperclip/Storage/S3
Lien 150 : http://railscasts.com/episodes/134-paperclip
Lien 151 : https://github.com/markevans/dragonfly
Lien 152 http://www.synbioz.com/blog/2011/12/28/le_kit_du_bon_developpeur_rails_partie_2
Lien 153 : http://synbioz.developpez.com/tutoriels/ruby/kit-developpeur-rail-2/
Lien 154 : http://www.w3.org/TR/html5/interactive-elements.html#the-menu-element
Lien 155 : http://dmouronval.developpez.com/menu/
Lien 156 http://blog.developpez.com/web/p10804/web/personnalisez-le-menu-contextuel-avec-fi/

Lien 157 : http://expressjs.com/
Lien 158 : http://socket.io/
Lien 159 : http://nodejs.developpez.com/tutoriels/javascript/nodejs-debuter-1/#LIV
Lien 160 : http://static.springsource.org/spring/docs/3.0.x/reference/mvc.html
Lien 161 : http://nodester.com/
Lien 162 : https://github.com/rmat0n/node-example/
Lien 163 : http://www.web-tambouille.fr/2011/03/8/node-js-partie-3-premiere-application-node-js-et-html5-express-socket-io.html
Lien 164 : http://www.web-tambouille.fr/
Lien 165 : http://nodejs.developpez.com/tutoriels/javascript/debuter-avec-node-js-partie-3/