Developpez Le Mag

Édition de juin – juillet 2014
Numéro 52
Magazine en ligne gratuit

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

Sommaire

Éditorial

Comme de coutume en cette période estivale, la rédaction se plie en quatre pour vous fournir le meilleur des publications de Developpez.com au format magazine que vous pouvez emmener sur la plage. Profitez-en bien et bonne vacances à tous !

La rédaction

Article CRM

Tester son code Apex

Comment tester son développement Apex avant une mise en production dans Salesforce ?
par Aurélien Laval
Page 16

Article Réseau

La technologie Ethernet

Ethernet est une technologie universelle qui dominait déjà les réseaux locaux bien avant le développement de Internet. La clé de la longévité de cette technologie, c'est sa simplicité. Souvent critiquée, elle a toujours été plus facile à utiliser et à mettre en œuvre que ses concurrentes.
par Philippe Latu
Page 35

OpenOffice-LibreOffice

Les derniers tutoriels et articles

Utiliser oBasic dans le Tableur (Calc)

Ce tutoriel ne sera qu'un élément de base pour la programmation, il ne contiendra pas toutes les commandes possibles, mais l'essentiel pour commencer à travailler avec le Tableur (Calc).

Je ne vais pas chercher à faire au cours de ce tutoriel un comparatif entre oBasic et VBA.

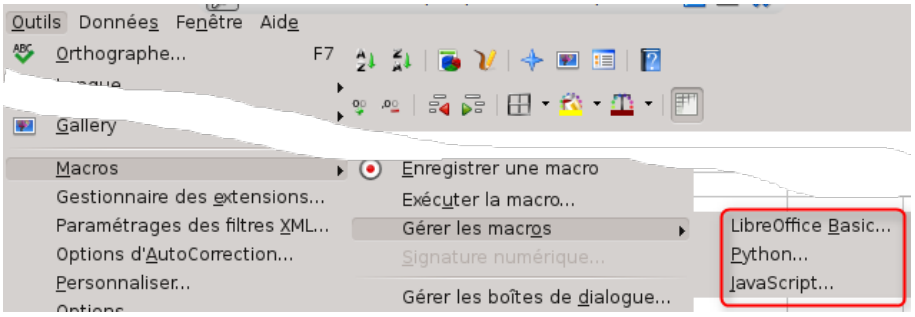
1 Introduction

1.1 Qu'est-ce qu'une macro ?

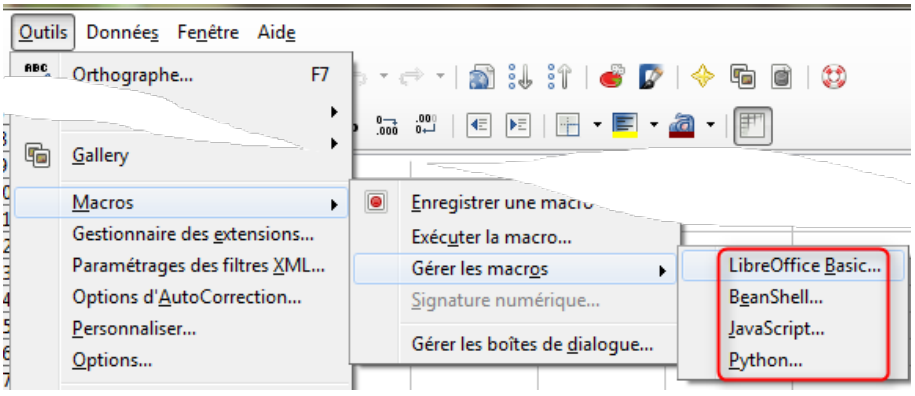
Une macro est une suite d'instructions dans un langage qui servira à les exécuter. Une macro est souvent utilisée pour automatiser des tâches répétitives ou longues.

Les macros pourront être écrites en **oBasic**, BeanShell, JavaScript ou Python. Toutefois, tout dépendra de votre environnement et de ce qui est installé par défaut sur le poste. Voici des captures sous différents environnements.

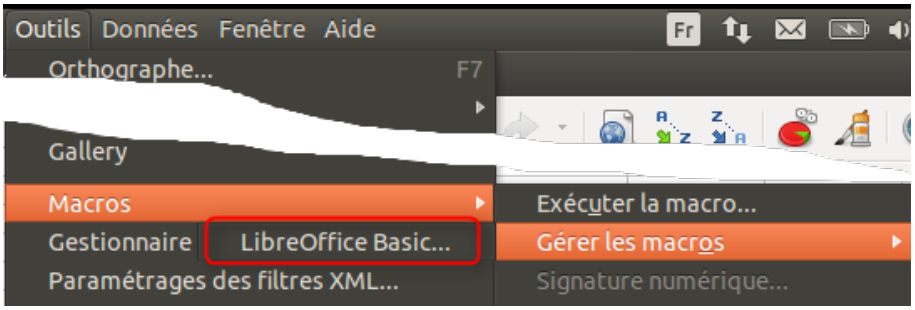
— Sous Windows :



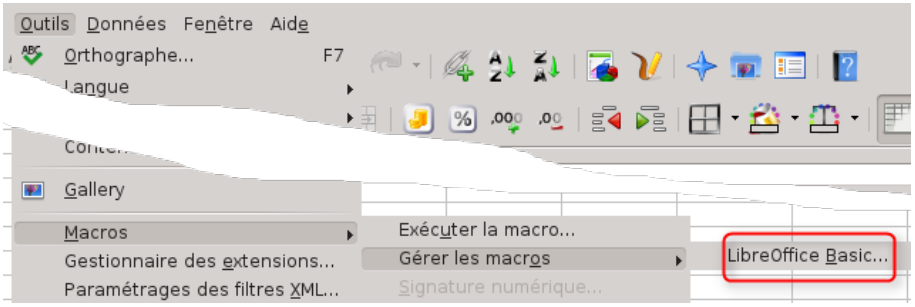
— Sous Ubuntu :



— Sous OpenSuse :



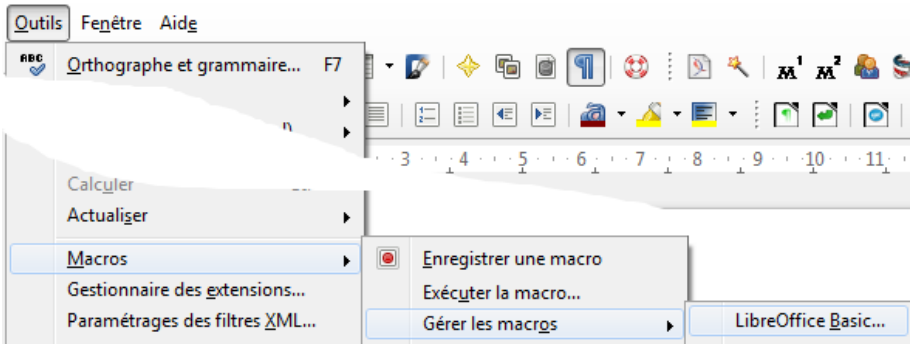
— Sous Debian :



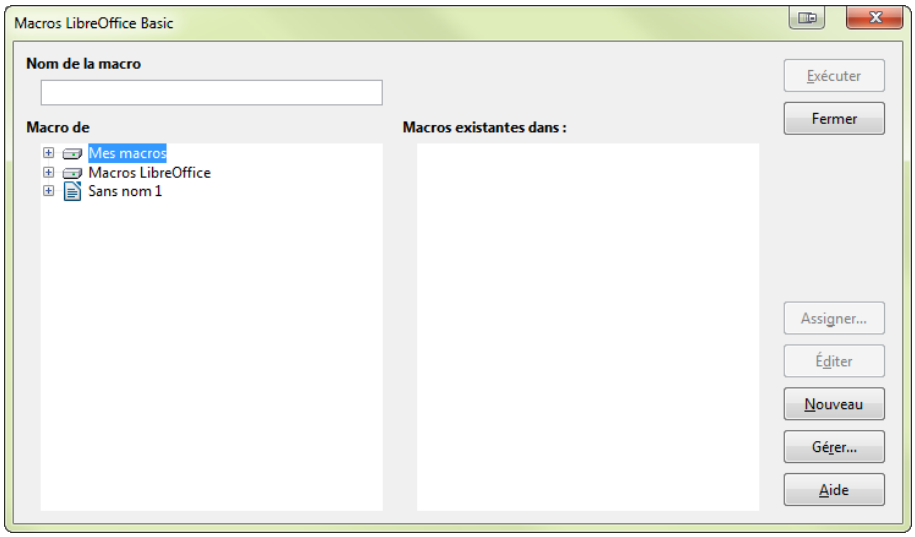
Comme vous pouvez le voir avec les captures précédentes, cela varie d'une distribution à une autre, mais dans ce tutoriel, je n'aborderai que la notion du **oBasic**, qui est commune à tous.

1.2 Où trouver l'éditeur de macros ?

L'éditeur de macro est accessible par la commande « Outils », « Macros », « Gérer les macros » et « LibreOffice Basic... » :



La fenêtre suivante apparaît :



Dans la partie gauche de la fenêtre se trouvent les emplacements de macros existantes :

— « Mes macros » contiendra toutes les macros qui seront rattachées au profil de la personne qui est connectée ;

— « Macro LibreOffice » contiendra toutes les macros qu'il sera possible d'utiliser avec la suite ;

— « Sans nom1 » (ou le nom d'un fichier) contiendra toutes les macros qu'il sera possible d'utiliser avec le fichier (si vous envoyez le fichier à quelqu'un, les macros contenues dans le fichier pourront être utilisées).

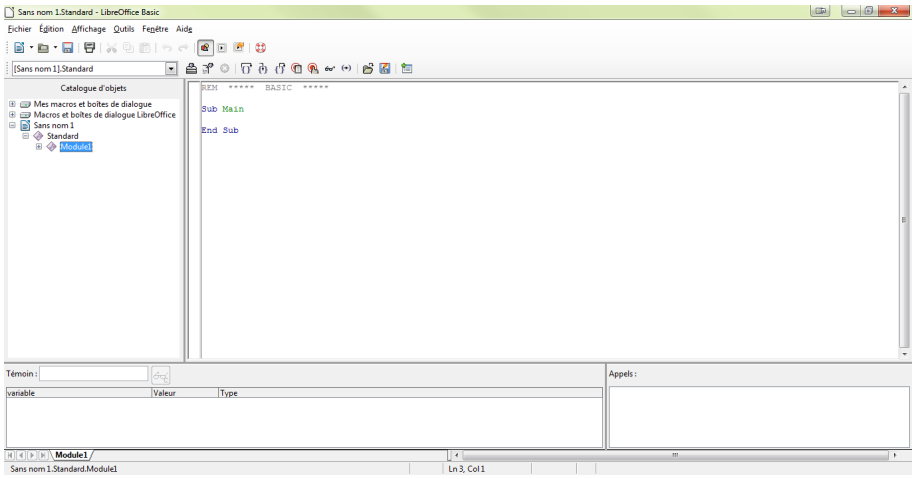
Dans la partie droite, c'est la liste des macros qui sont disponibles.

Pour ouvrir l'éditeur, deux cas possibles suivant où vous vous trouvez :

— soit vous voulez modifier ou compléter une macro, dans ce cas, il faudra cliquer sur « Gérer » et sélectionner la macro correspondante ;

— soit vous voulez créer une nouvelle macro et dans ce cas cliquer sur « Nouveau ».

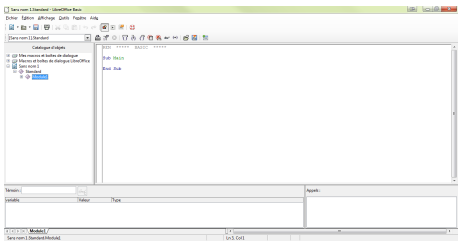
La fenêtre de l'éditeur est la suivante :



L'exécution des macros est bloquée par défaut, ceci afin d'éviter des programmes indésirables ou des virus.
Vous trouverez ici une procédure pour activer correctement les macros.

2 L'éditeur oBasic

L'éditeur oBasic se présente comme toute application :



La fenêtre peut être décomposée en quatre parties :

— la première partie contient toutes les commandes (menus, barres d'outils) ;

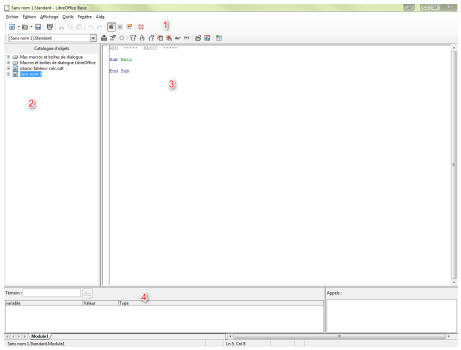
— la deuxième contient les catalogues ;

— la troisième contient le code ;

— et la quatrième contient les témoins.

2.1 Menus et barres d'outils

Elle contient, entre autres, toutes les fonctions d'édition standard aux suites, mais aussi des barres d'outils, comme celle-ci, dont nous nous servirons souvent :



— (1) permet de sélectionner le module souhaité ;

— (2) compile le code et permet d'identifier les erreurs ;

— (3) exécute la macro sélectionnée ;

— (4) arrête une macro en cours, il est aussi possible de faire Maj+Ctrl+S ;

— (5) exécute une procédure et s'arrête ;

— (6) exécute l'étape et s'arrête à l'exécution suivante ;
— (7) retourne la précédente macro active ;
— (8) insère un point d'arrêt ;
— (9) gère les points d'arrêt avec une boîte de dialogue ;
— (10) active ou désactive le témoin ;
— (11) met en évidence le texte entre deux parenthèses ;
— (12) ouvre le texte source oBasic dans la fenêtre ;
— (13) enregistre le code ;
— (14) permet d'importer une boîte de dialogue.

2.2 Les catalogues

Un catalogue est l'emplacement où seront stockées les différentes macros. Il en existe trois types :

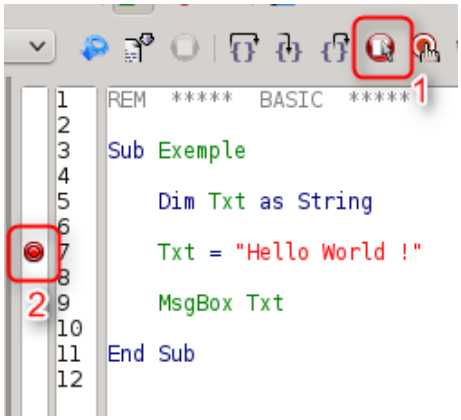
— « Mes macros et boîtes de dialogue » : il s'agit du catalogue contenant vos macros, celles-ci seront alors disponibles pour toutes les applications ;
— « Macros et boîtes de dialogue... » : il s'agit du catalogue dédié aux macros fournies avec LibO et AOO ou les extensions ;
— le catalogue du document chargé, mais la macro ne sera alors disponible qu'avec le fichier.

2.3 Le code

Cette partie contiendra tout le code. C'est dans cette zone que nous pourrons saisir de nouveaux codes ou modifier les codes existants.

Les différents codes seront identifiés par des noms de fonctions (nous en reparlerons par la suite), ce qui nous permettra de nous déplacer facilement d'une fonction à une autre pour modifier ou rajouter du code dedans.

Il sera aussi possible dans cette zone de visualiser les points d'arrêt. Un point d'arrêt permet d'arrêter le déroulement d'une macro sans la stopper au niveau de l'opération qui se trouve juste avant. Par exemple :



Il vous suffira de vous mettre sur la ligne 7, de cliquer sur la commande (1) et le point rouge (2) apparaîtra sur la ligne. Si vous lancez votre macro, toutes les commandes qui précèdent les points sont exécutées.

Les points d'arrêt sont très utiles quand vous devez chercher une erreur dans votre code.

2.4 Le témoin

Cette partie permet d'observer la valeur des variables au cours de l'exécution du code.

3 Quelques principes

Comme dit précédemment, une macro est un bout de code écrit, dans notre cas, en **oBasic**. Elle exécutera une suite d'opérations dans le fichier Tableur.

Voici, au travers de quelques exemples, ce qu'il est possible de faire...

Exemple 1, nous affichons un message :

```
1  Sub Exemple_1
2
3      Dim Txt as String
4
5      Txt = "Hello World !"
6
7      MsgBox Txt
8
9  End Sub
```

Nous obtiendrons donc :

Exemple 2, nous affichons un message que nous aurons saisi :

```
1  Sub Exemple_2
2
3      Dim Txt as String
4
5      Txt = InputBox ("Titre")
6
7      MsgBox Txt
8
9  End Sub
```

La fenêtre suivante apparaît :

Nous saisissons le texte souhaité :

Nous obtenons donc :

Dans les deux exemples précédents, nous avons utilisé les balises `Sub`, mais nous aurions pu utiliser les balises `Function`.
Les deux codes fonctionnent également avec les autres applications de la suite bureautique.

Nous venons de voir comment se compose une macro :

- une balise d'ouverture et une de fermeture ;
- la déclaration des variables, cette étape est importante, car elle est souvent source d'erreurs ;
- le contenu de la macro qui décrit les différentes opérations à réaliser.

4 Onglets

4.1 Lister les onglets présents dans le classeur

Voici le code qui nous permettra de lister tous les onglets contenus dans le classeur :

```
1  Sub ListerOnglet()
2   Dim monDocument As Object
3   Dim lesFeuilles as Object
4   Dim uneFeuille As Object
5   Dim nbF as Integer
6
7      monDocument = ThisComponent
8      lesFeuilles = monDocument.Sheets
9      nbF = lesFeuilles.Count
10
11     For i = 0 to nbF-1
12       uneFeuille = lesFeuilles(i)
13       MsgBox uneFeuille.Name
14     Next
15
16 End Sub
```

Explications :

- il est recommandé de définir toutes les variables, cela permet d'en connaître le type pour la suite :

  Très utile, quand le code est long. Vous pourrez ainsi vous souvenir du type que vous aviez associé à la variable.

  Si vous ne la définissez pas, la valeur par défaut est Object.

- `monDocument = ThisComponent` :
  permet d'indiquer que nous allons travailler sur le document. Nous aurions pu mettre monDocument = `CurrentComponent`, cette première notation nous indique que c'est le document actif qui est utilisé. La seconde notation permet d'intervenir sur un autre document ;
- `lesFeuilles = monDocument.Sheets` :
  permet d'indiquer que nous allons travailler sur les différentes feuilles de l'onglet ;
- `lesFeuilles.Count` :
  donne le nombre de feuilles que contient le classeur ;
- `For i = 0 to nbF-1 ... Next` :
  va nous permettre de balayer toutes les feuilles :

  Attention, le premier onglet correspond à la valeur 0, et il ne faudra pas oublier de retirer 1 au nombre total, sinon vous obtiendrez une belle erreur.

- `uneFeuille = lesFeuilles(i)` :
  permet de sélectionner une feuille ;
- `MsgBox uneFeuille.Name` :
  nous donne le nom de la feuille.

Créez un nouveau classeur, insérez-y différents onglets, copiez le code dans une nouvelle macro et exécutez-la.

4.2 Quelques commandes

4.2.a Sélectionner une feuille par son nom

```
Sub SelectionnerFeuille()
 Dim monDocument As Object
 Dim lesFeuilles as Object
 Dim uneFeuille As Object

    monDocument = ThisComponent
    lesFeuilles = monDocument.Sheets
    uneFeuille = lesFeuilles.getByName("
        Feuille1")

    ...

End Sub
```

4.3 Détecter la feuille active

```
Sub FeuilleActive()

    ...

    Msgbox ThisComponent.
        CurrentController.ActiveSheet.
        Name

    ...

End Sub
```

4.4 Renommer une feuille

```
Sub RenommerFeuille()
 Dim monDocument As Object
 Dim lesFeuilles as Object
 Dim uneFeuille As Object

    monDocument = ThisComponent
    lesFeuilles = monDocument.Sheets
    uneFeuille = lesFeuilles.getByName("
        Feuille1")
    uneFeuille.Name = "Feuille renommée"

    ...

End Sub
```

4.5 Insérer/supprimer une feuille

```
Sub Inserer_Supprimer()
 Dim monDocument As Object
 Dim lesFeuilles as Object
 Dim uneFeuille As Object
 Dim indexFeuille As Integer

    monDocument = ThisComponent
    lesFeuilles = monDocument.Sheets

    'Pour insérer une feuille après la
        feuille sélectionnée
    uneFeuille = lesFeuilles.getByName("
        Feuille1")
    indexFeuille = uneFeuille.
        RangeAddress.Sheet
    lesFeuilles.insertNewByName("Feuille
         insérée", indexFeuille+1)

    'Pour supprimer une feuille
    lesFeuilles.removeByName("Feuille1")


End Sub
```

5 Cellules

Une des particularités du Tableur de LibreOffice (ou OpenOffice) est que vous ne pourrez pas sélectionner une colonne ou une ligne, vous serez obligé de sélectionner une zone.

5.1 Sélectionner une cellule

5.1.a Une cellule

Il existe trois façons de sélectionner une cellule :

```
Sub SelectionnerCellule()
 Dim monDocument As Object
 Dim lesFeuilles as Object
 Dim maFeuille As Object
 Dim maCellule As Object

    monDocument = ThisComponent
    lesFeuilles = monDocument.Sheets
    maFeuille = lesFeuilles.getByName("
        Feuille1")

    'Par son adresse
    maCellule = maFeuille.
        getCellRangeByName("A10")

    'Par son nom (si vous avez donné un
        nom à une cellule)
    maCellule = maFeuille.
        getCellRangeByName("Cellule_nomm
        ée")

    'Par sa position (colonne, ligne)
    maCellule = maFeuille.
        getCellRangeByPosition(1,10)

End Sub
```

5.1.b Une zone

```
Sub SelectionnerZone()
 Dim monDocument As Object
 Dim lesFeuilles as Object
 Dim maFeuille As Object
 Dim maCellule As Object

    monDocument = ThisComponent
    lesFeuilles = monDocument.Sheets
    maFeuille = lesFeuilles.getByName("
        Feuille1")

    'Par son adresse
    maCellule = maFeuille.
        getCellRangeByName("A1:A10")

    'Par son nom (si vous avez donné un
        nom à une zone)
    maCellule = maFeuille.
        getCellRangeByName("Zone_nommée"
        )

    'Par sa position (colonne, ligne)
```

```
18      maCellule = maFeuille.
            getCellRangeByPosition(1,1,1,10)
19
20  End Sub
```

5.1.c La cellule active

```
1   Sub CelluleActive()
2    Dim maCellule As Integer
3
4       maCellule = ThisComponent.
            CurrentSelection
5       MsgBox maCellule.CellAddress.Column
6       MsgBox maCellule.CellAddress.Row
7       MsgBox maCellule.CellAddress.Sheet
8
9   End Sub
```

5.1.d La dernière cellule d'une ligne

```
1   Sub DerniereCellule()
2    Dim monDocument As Object
3    Dim lesFeuilles as Object
4    Dim maFeuille As Object
5    Dim maCellule As Object
6    Dim maPosition As Object
7
8       monDocument = ThisComponent
9       lesFeuilles = monDocument.Sheets
10      maFeuille = lesFeuilles.getByName("
            Feuille1")
11
12      maPosition = maFeuille.createCursor
13      maPosition.gotoEndOfUsedArea(False)
14      MsgBox maPosition.RangeAddress.
            EndRow+1
15      'le +1 permet de se positionner sur
            la dernière ligne vide
16
17  End Sub
```

5.2 Écrire dans une cellule

Il y a plusieurs façons d'écrire dans une cellule suivant ce que vous voulez y mettre.

5.2.a Une valeur numérique

```
1   Sub EcrireValeur()
2    Dim monDocument As Object
3    Dim lesFeuilles as Object
4    Dim maFeuille As Object
5    Dim maCellule As Object
6
7       monDocument = ThisComponent
8       lesFeuilles = monDocument.Sheets
9       maFeuille = lesFeuilles.getByName("
            Feuille1")
10      maCellule = maFeuille.
            getCellRangeByName("A1")
11      maCellule.Value = 1
12      'Vous pouvez y mettre aussi un
            calcul
13
14  End Sub
```

5.2.b Du texte

```
1   Sub EcrireTexte()
2    Dim monDocument As Object
3    Dim lesFeuilles as Object
4    Dim maFeuille As Object
5    Dim maCellule As Object
6
7       monDocument = ThisComponent
8       lesFeuilles = monDocument.Sheets
9       maFeuille = lesFeuilles.getByName("
            Feuille1")
10      maCellule = maFeuille.
            getCellRangeByName("A1")
11      maCellule.String = "Texte que vous
            voulez mettre dans la cellule"
12
13  End Sub
```

5.2.c Une formule

```
1   Sub EcrireFormule()
2    Dim monDocument As Object
3    Dim lesFeuilles as Object
4    Dim maFeuille As Object
5    Dim maCellule As Object
6
7       monDocument = ThisComponent
8       lesFeuilles = monDocument.Sheets
9       maFeuille = lesFeuilles.getByName("
            Feuille1")
10      maCellule = maFeuille.
            getCellRangeByName("A1")
11
12      'Pour une formule en anglais
13      maCellule.Formula = "=SUM(A1:A100")
14
15      'Pour une formule avec la langue du
            poste en français
16      maCellule.FormulaLocal = "=SOMME(A1:
            A100")
17
18  End Sub
```

5.3 Lire une cellule

Tout comme pour écrire dans une cellule, la lecture est soumise aux mêmes conditions.

Pour les exemples qui vont suivre, nous allons saisir les données suivantes dans un classeur :

- A1 contient la valeur 1 ;
- A2 contient le texte « `Test` » ;
- A3 contient la formule « `=SOMME(B1:B6)` » (dans mon cas la somme fera 21).

Il n'y aura aucun message d'erreur si vous vous trompez sur la commande. Car la valeur que vous obtenez est « 0 ».

5.3.a Une valeur

```
1   Sub EcrireFormule()
2    Dim monDocument As Object
3    Dim lesFeuilles as Object
4    Dim maFeuille As Object
5    Dim maCellule As Object
6
7       monDocument = ThisComponent
8       lesFeuilles = monDocument.Sheets
9       maFeuille = lesFeuilles.getByName("
            Feuille1")
10
```

```
11      maCellule = maFeuille.
            getCellRangeByName("A1")
12      Msgbox maCellule.Value
13
14      maCellule = maFeuille.
            getCellRangeByName("A2")
15      Msgbox maCellule.Value
16
17      maCellule = maFeuille.
            getCellRangeByName("A3")
18      Msgbox maCellule.Value
19
20  End Sub
```

Ce qui nous donne pour les différentes Msgbox :

- 1 ;
- 0 ;
- 21.

5.3.b Du texte

```
1   Sub LireTexte()
2    Dim monDocument As Object
3    Dim lesFeuilles as Object
4    Dim maFeuille As Object
5    Dim maCellule As Object
6
7       monDocument = ThisComponent
8       lesFeuilles = monDocument.Sheets
9       maFeuille = lesFeuilles.getByName("
            Feuille1")
10      maCellule = maFeuille.
            getCellRangeByName("A1")
11      Msgbox maCellule.String
12
13      maCellule = maFeuille.
            getCellRangeByName("A2")
14      Msgbox maCellule.String
15
16      maCellule = maFeuille.
            getCellRangeByName("A3")
17      Msgbox maCellule.String
18
19  End Sub
```

Ce qui nous donne pour les différentes Msgbox :

- 1 ;
- Test ;
- 21.

5.3.c Une formule

```
1   Sub LireFormule()
2    Dim monDocument As Object
3    Dim lesFeuilles as Object
4    Dim maFeuille As Object
5    Dim maCellule As Object
6
7       monDocument = ThisComponent
8       lesFeuilles = monDocument.Sheets
9       maFeuille = lesFeuilles.getByName("
            Feuille1")
10      maCellule = maFeuille.
            getCellRangeByName("A1")
11      Msgbox maCellule.Formula
12
13      maCellule = maFeuille.
            getCellRangeByName("A2")
14      Msgbox maCellule.Formula
15
16      maCellule = maFeuille.
            getCellRangeByName("A3")
17      Msgbox maCellule.Formula
18
19  End Sub
```

Ce qui nous donne pour les différentes Msgbox :

- 1 ;
- Test ;
- SUM(B1 :B6).

Si vous aviez mis `FormulaLocal`, le résultat aurait été `=SOMME(B1:B6)`.

5.4 Effacer le contenu d'une cellule

La suppression agit de la même façon que la lecture, elle tient compte du type. Nous allons conserver les mêmes données qu'avant :

- A1 contient la valeur 1 ;
- A2 contient le texte « `Test` » ;
- A3 contient la formule « `=SOMME(B1:B6)` ».

5.4.a Une valeur

```
1   Sub EffacerValeur()
2    Dim monDocument As Object
3    Dim lesFeuilles as Object
4    Dim maFeuille As Object
5    Dim maCellule As Object
6    Dim aEffacer As Long
7
8       monDocument = ThisComponent
9       lesFeuilles = monDocument.Sheets
10      maFeuille = lesFeuilles.getByName("
            Feuille1")
11
12      aEffacer = com.sun.star.sheet.
            CellFlags.VALUE
13
14      maCellule = maFeuille.
            getCellRangeByName("A1")
15      maCellule.clearContents(aEffacer)
16
17      maCellule = maFeuille.
            getCellRangeByName("A2")
18      maCellule.clearContents(aEffacer)
19
20      maCellule = maFeuille.
            getCellRangeByName("A3")
21      maCellule.clearContents(aEffacer)
22
23  End Sub
```

Ce qui nous donne pour les différentes cellules :

- A1 ne contient plus aucune donnée ;
- A2 contient « `Test` » ;
- A3 contient « `=SUM(B1:B6)` ».

5.4.b Du texte

```
Sub EffacerTexte()
 Dim monDocument As Object
 Dim lesFeuilles as Object
 Dim maFeuille As Object
 Dim maCellule As Object

    monDocument = ThisComponent
    lesFeuilles = monDocument.Sheets
    maFeuille = lesFeuilles.getByName("
        Feuille1")

    aEffacer = com.sun.star.sheet.
        CellFlags.STRING

    maCellule = maFeuille.
        getCellRangeByName("A1")
    maCellule.clearContents(aEffacer)

    maCellule = maFeuille.
        getCellRangeByName("A2")
    maCellule.clearContents(aEffacer)

    maCellule = maFeuille.
        getCellRangeByName("A3")
    maCellule.clearContents(aEffacer)

End Sub
```

Ce qui nous donne pour les différentes cellules :

- A1 contient « `1` » ;
- A2 ne contient plus aucune donnée ;
- A3 contient « `=SUM(B1:B6)` ».

5.4.c Une formule

```
Sub EffacerFormule()
 Dim monDocument As Object
 Dim lesFeuilles as Object
 Dim maFeuille As Object
 Dim maCellule As Object

    monDocument = ThisComponent
    lesFeuilles = monDocument.Sheets
    maFeuille = lesFeuilles.getByName("
        Feuille1")

    aEffacer = com.sun.star.sheet.
        CellFlags.FORMULA

    maCellule = maFeuille.
        getCellRangeByName("A1")
    maCellule.clearContents(aEffacer)

    maCellule = maFeuille.
        getCellRangeByName("A2")
    maCellule.clearContents(aEffacer)

    maCellule = maFeuille.
        getCellRangeByName("A3")
    maCellule.clearContents(aEffacer)

End Sub
```

Ce qui nous donne pour les différentes cellules :

- A1 contient « `1` » ;
- A2 contient « `Test` » ;
- A3 ne contient plus aucune donnée ;

5.4.d Tout supprimer

Il existe une façon de tout effacer d'un seul coup :

```
Sub EffacerTout()
 Dim monDocument As Object
 Dim lesFeuilles as Object
 Dim maFeuille As Object
 Dim maCellule As Object

    monDocument = ThisComponent
    lesFeuilles = monDocument.Sheets
    maFeuille = lesFeuilles.getByName("
        Feuille1")

    aEffacer = com.sun.star.sheet.
        CellFlags.VALUE +_
                com.sun.star.sheet.
                    CellFlags.STRING +_
               com.sun.star.sheet.
                   CellFlags.FORMULA

    maCellule = maFeuille.
        getCellRangeByName("A1")
    maCellule.clearContents(aEffacer)

    maCellule = maFeuille.
        getCellRangeByName("A2")
    maCellule.clearContents(aEffacer)

    maCellule = maFeuille.
        getCellRangeByName("A3")
    maCellule.clearContents(aEffacer)

End Sub
```

Toutes les cellules A1, A2 et A3 sont maintenant vides.

6 Conclusion

Nous venons de voir quelques-unes des fonctions qui vous permettront de commencer la programmation avec **oBasic**. La liste des fonctions est longue, vous pouvez toutes les découvrir dans cet ouvrage : Programmation OpenOffice.org et LibreOffice.

À vous maintenant de vous lancer dans la programmation et de développer vos propres applications.

Retrouvez l'article de ***Vincent Viale*** *en ligne : lien 6*

Apprendre à créer un diagramme de Gantt avec Tableur

Je vais vous apprendre au travers de ce tutoriel comment réaliser un diagramme de Gantt avec Tableur (Calc).

1 Définition

Le **diagramme de Gantt** est un outil utilisé en gestion de projet, il permet de visualiser dans le temps les diverses tâches composant un projet par une représentation graphique de l'avancement du projet. Mais vous verrez que cette représentation peut aussi servir dans d'autres cas.

2 Les données

Un **diagramme de Gantt** doit contenir certaines données, voici un exemple simpliste des données nécessaires :

	A	B	C	D	E
1	Étapes	Date de début	Date de fin	Réalisé	Restant à faire
2	Phase 1	18/11/2013	12/12/2013	5	19
3	Phase 2	19/11/2013	06/12/2013	2	15
4	Phase 3	20/11/2013	26/11/2013		6
5	Phase 4	23/11/2013	24/11/2013		1
6	Phase 5	26/11/2013	12/12/2013	1	15
7	Phase 6	27/11/2013	31/12/2013	2	32

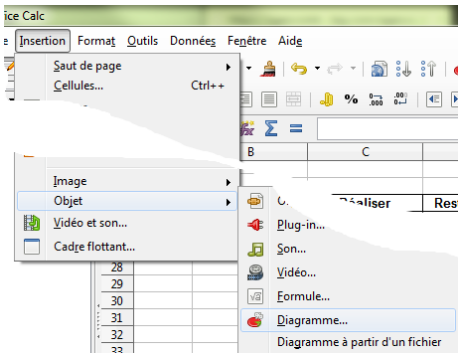
Dans notre cas, pour simplifier les calculs, j'ai pris comme hypothèse que le « Restant à faire » était la différence entre la date de fin et la date de début moins le réalisé. *Je n'ai fait aucun filtre sur les jours ouvrés ou pas.*

Mais pour créer le diagramme, nous n'avons pas besoin de toutes les données, voici celles dont nous avons besoin, il nous suffit de réorganiser le tableau en mettant la date de fin à la fin du tableau, mais nous ne nous en servirons pas dans le diagramme :

Étapes	Date de fin	Réalisé	Restant à faire	Date de fin
Phase 1	18/11/13	5	19	12/12/13
Phase 2	19/11/13	2	15	06/12/13
Phase 3	20/11/13	0	6	26/11/13
Phase 4	23/11/13	0	1	24/11/13
Phase 5	26/11/13	1	15	12/12/13
Phase 6	27/11/13	2	32	31/12/13

3 Création du graphique

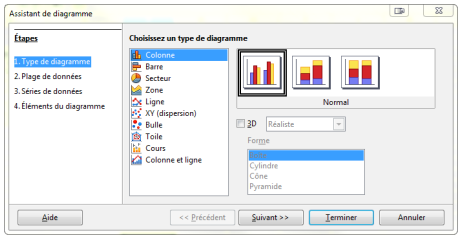
La commande se trouve dans le menu « Insertion », « Objet » et « Diagramme » :



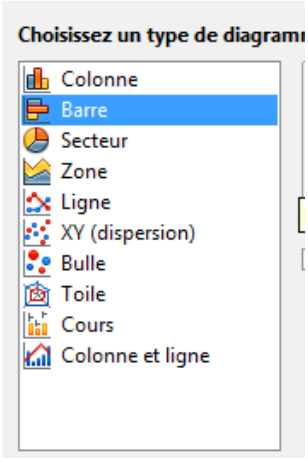
Ou dans la barre de menu :

3.1 Sélection du graphique

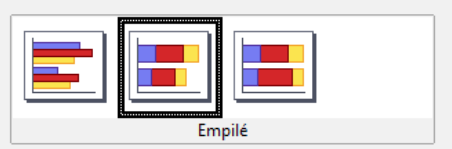
La fenêtre suivante apparaît :



Il nous faut sélectionner le type de diagramme « Barre » :



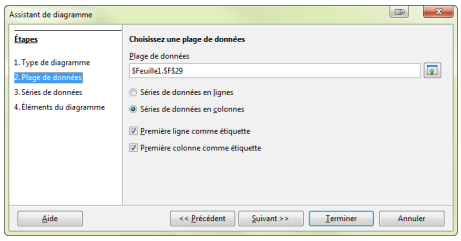
Ensuite, nous sélectionnons « Empilé » :



Il ne nous reste plus qu'à cliquer sur « Suivant ».

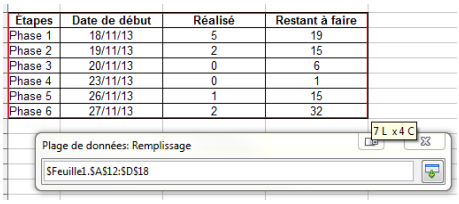
3.2 Sélection des données

Nous arrivons sur la fenêtre suivante :



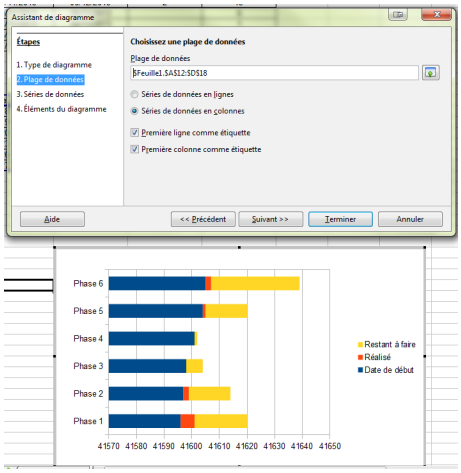
Cliquons sur :

Ensuite, sélectionnons les données :

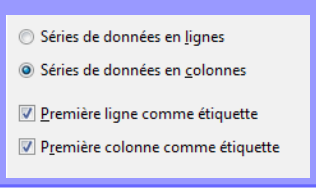


Étapes	Date de début	Réalisé	Restant à faire
Phase 1	18/11/13	5	19
Phase 2	19/11/13	2	15
Phase 3	20/11/13	0	6
Phase 4	23/11/13	0	1
Phase 5	26/11/13	1	15
Phase 6	27/11/13	2	32

Ce qui nous donne finalement :

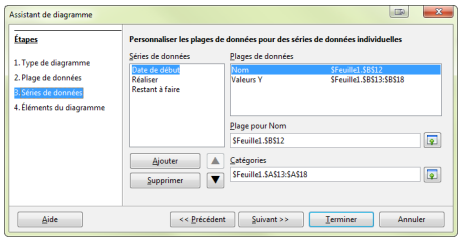


Je laisse le soin à chacun de modifier ses paramètres en fonction de ses données :



3.3 Contrôle des données

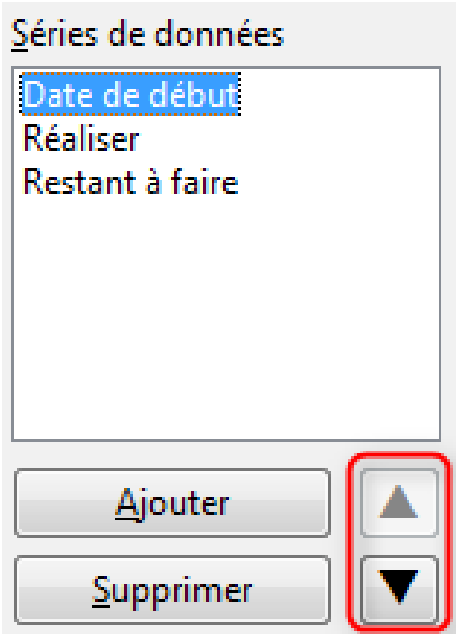
La fenêtre suivante apparaît :



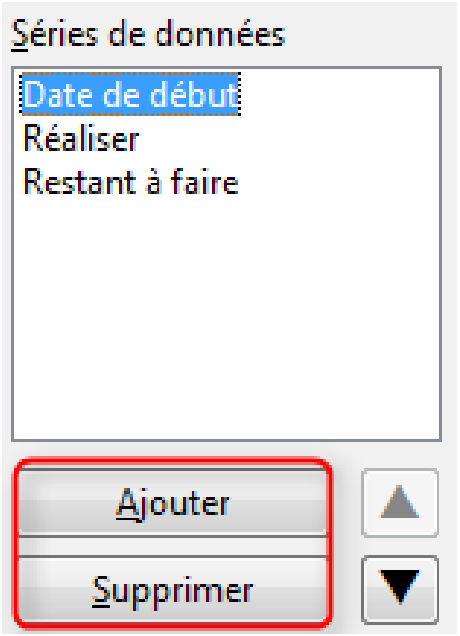
Elle nous permet de faire un contrôle des données, et nous permet aussi de modifier certaines informations. Comme :

— le titre d'une donnée, il nous suffirait de modifier la donnée :

— de déplacer l'ordre des données, il suffit de sélectionner la donnée et de la mettre à l'emplacement voulu :



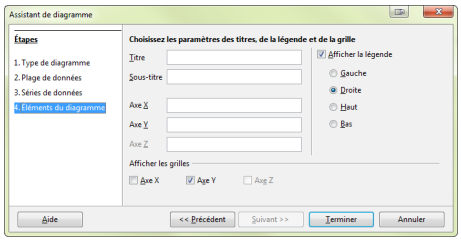
— d'ajouter ou supprimer des données :



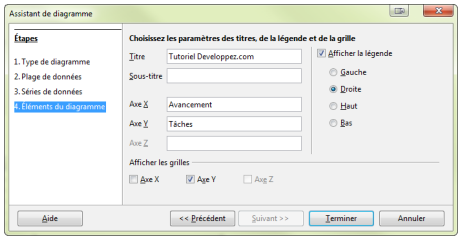
Une fois toutes les modifications faites, il ne nous reste plus qu'à cliquer sur « Suivant ».

3.4 Information du diagramme

La fenêtre suivante apparaît :



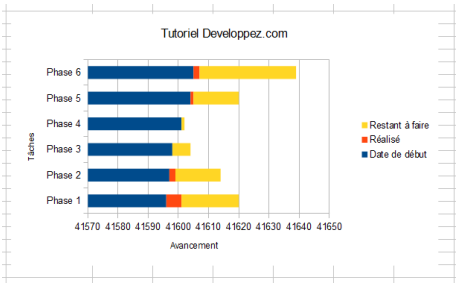
Cette fenêtre contient les éléments textuels visibles sur les diagrammes, pour notre exemple, je vous propose :



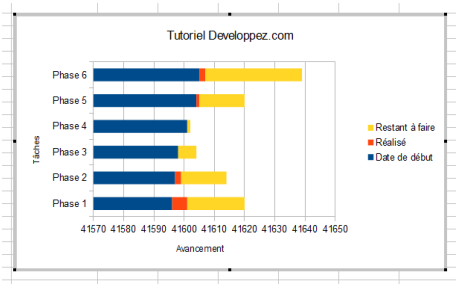
Il ne nous reste plus qu'à cliquer sur « Terminer ».

3.5 Mise en forme du diagramme de Gantt

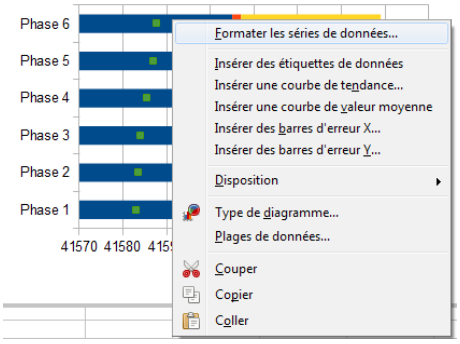
Nous obtenons avec les étapes précédentes le graphique suivant :



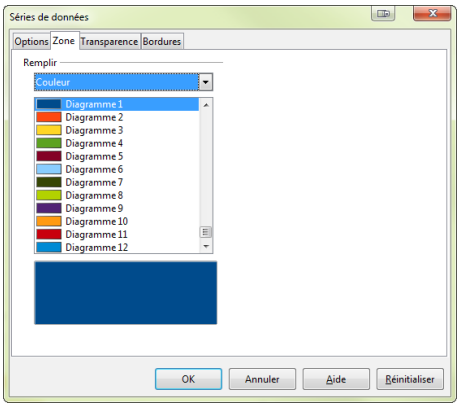
Maintenant, il nous faut modifier le diagramme, pour cela faites un double-clic sur le graphique, nous obtenons ceci :



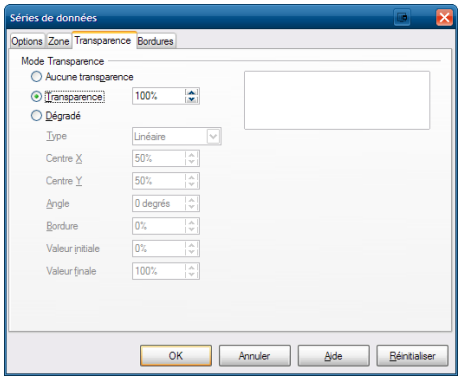
Il nous faut sélectionner les données correspondant aux données de la date de début, il suffit de se mettre dessus et faire un clic droit :



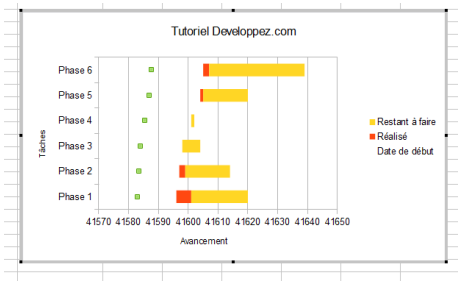
Il nous faut sélectionner « Formater les séries de données... », la fenêtre suivante s'ouvre :



Il nous faut sélectionner l'onglet « Transparence », et mettre la transparence à 100

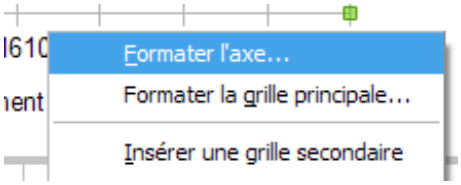


Ce qui nous donne maintenant :

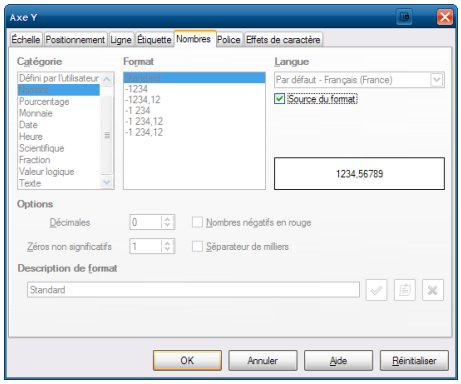


Si les couleurs ne vous plaisent pas, elles sont modifiables dans l'onglet « Zone ».

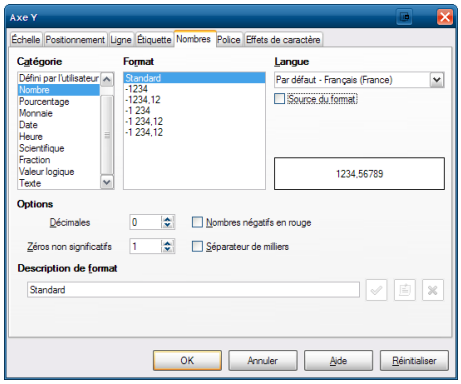
Maintenant, mettons à jour la barre des x, il faut pour cela faire un clic droit sur l'axe, et sélectionner « Formater l'axe... » :



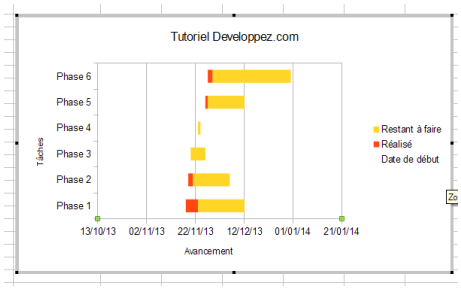
La fenêtre suivante s'ouvre :



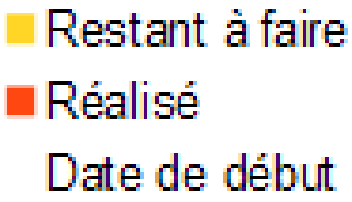
Dans l'onglet « Nombres », tout est grisé, il faut décocher « Source du format » :



Il ne nous reste plus qu'à sélectionner le format souhaité, dans notre cas, il nous faut un format de date, ce qui nous donnera donc :



Maintenant, il ne nous reste plus qu'à supprimer la légende « Date de début » :



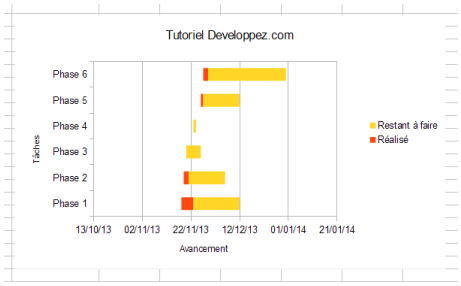
Malheureusement, la légende n'est pas modifiable, mais nous avons quand même deux façons de faire :

— soit en redimensionnant le cadre de la légende de façon à ce que la dernière ligne n'y loge pas :

— soit en saisissant un espace dans le nom de la colonne correspondante :

Étapes		Réalisé	Restant à faire
Phase 1	18/11/13	5	19
Phase 2	19/11/13	2	15
Phase 3	20/11/13	0	6
Phase 4	23/11/13	0	1
Phase 5	26/11/13	1	15
Phase 6	27/11/13	2	32

Finalement, nous obtenons le diagramme suivant :



4 Conclusion et remerciements

Nous venons de voir comment réaliser un diagramme de **Gantt** avec Tableur, maintenant à vous de voir comment vous voulez l'utiliser.

Retrouvez l'article de ***Vincent Viale*** *en ligne : lien 7*

CRM

Les derniers tutoriels et articles

Tester son code Apex

Comment tester son développement Apex avant une mise en production dans Salesforce ?

1 Introduction

Lors d'une mise en production d'une fonctionnalité Salesforce ayant nécessité un développement Apex (trigger, contrôleur ?), il est obligatoire que celle-ci ait une couverture de code d'au moins 75 % et que l'exécution de cette couverture ne provoque aucune erreur (non traitée du moins) pour que la mise en production soit validée par Salesforce.

Avant d'aller plus loin, je vous propose quelques rappels sur les termes qui vont être abordés tout au long de ce tutoriel :

- Salesforce lien 8 est un CRM (Customer Relationship Manager) orienté SaaS (Software as a Service : lien 9) et PaaS (Platform as a Service lien 10) permettant la gestion des relations clients d'une entreprise ;
- Apex, est un langage de programmation côté serveur permettant de modifier la logique métier et le traitement des données ;
- un trigger lien 11 est un morceau de code Apex qui s'exécute avant et/ou après :
  - une création d'enregistrements,
  - une mise à jour d'enregistrements,
  - une suppression d'enregistrements ;
- WSDL (Web Services Description Language : lien 12) est, comme son nom l'indique, un fichier au format XML décrivant le web service, c'est-à-dire les méthodes que vous pouvez appeler avec les paramètres attendus, la valeur de retour ainsi que les types de ces valeurs et l'URL d'appel de ce web service pour faire simple.

Maintenant que vous avez connaissance des définitions, je vais vous montrer comment créer une classe de tests qui va nous servir à couvrir notre code, mais également comment vous assurer que le résultat retourné correspond bien à celui attendu. Pour cela, nous allons, dans un premier temps, écrire un trigger qui se déclenchera lors de la création et de la mise à jour d'enregistrements de type « Case », pour ensuite créer sa classe de tests unitaires. Je finirai par vous expliquer comment tester l'appel d'un service web en simulant les données retournées par celui-ci.

Le scénario est assez simple, lorsqu'un enregistrement de type « Case » sera créé ou mis à jour, nous le rattacherons au contact correspondant à l'adresse mail de cet enregistrement, si celui-ci n'est pas vide. Dans le cas où l'adresse mail est vide ou que le contact n'existe pas, nous allons le créer puis le rattacher au « Case » approprié.

2 Prérequis

Avant toute chose, vous devez disposer d'une instance afin de pouvoir suivre ce tutoriel. Si ce n'est pas le cas, je vous invite à vous rendre à cette adresse : lien 13. Salesforce met à disposition plusieurs types d'instances qui proposent différentes fonctionnalités et donc à différents prix (prix par licence utilisateur par mois). La seule instance que vous pourrez généreusement obtenir de la part de Salesforce est celle de développement (celle que je vous propose via le lien ci-dessus) qui, comme son nom l'indique, sert à tester et développer des services. Je vous laisse juger par vous-même la grille tarifaire sur les diverses instances que propose le CRM : lien 14.

Vous aurez également besoin d'un environnement de développement pour écrire votre code, vous pouvez utiliser celui fourni par Salesforce ou alors vous procurer celui compatible avec Eclipse et qui s'appelle Force.com IDE (je l'utiliserai tout au long de ce tutoriel). Pour ce dernier, rendez-vous sur l'Eclipse MarketPlace et installez-le.

3 Écriture du trigger

Comme je vous l'ai dit plus haut, un trigger est un bout de code Apex qui s'exécute avant et/ou après :

- la création d'un enregistrement ;
- la mise à jour d'un enregistrement ;
- la suppression d'un enregistrement.

À savoir que lorsque l'on écrit un trigger ou du code Apex de manière générale, Salesforce impose des limites que nous devons impérativement respecter pour que notre développement puisse être validé et déployé en production. Par exemple, nous ne devons pas excéder plus de 100 requêtes SOQL (Salesforce Object Query Language), il faut donc réfléchir en termes de List. Je veux dire par là que si, par exemple, le trigger se déclenche et qu'il a une cinquantaine d'enregistrements à traiter et que nous avons besoin de faire des requêtes vers la base de données, nous n'allons pas faire une requête par enregistrement en utilisant la condition « WHERE Id = '' », mais plutôt faire une seule requête en spécifiant à la condition un tableau contenant tous les identifiants que nous souhaitons, ce qui donnerait la condition « WHERE Id IN :monTableau ». Ainsi, nous ne faisons qu'une seule requête au lieu de n.

Salesforce a récemment imposé que toute transaction n'excède pas les 10 secondes de traitement CPU (du côté des serveurs Salesforce). Cela exclut les temps d'accès à la base de données et ne prend en compte uniquement que le traitement de données (triggers, méthodes de classes, etc.).

Vous pouvez visualiser le temps d'exécution d'un processus à l'aide de la méthode Limits.getCpuTime.

Maintenant que vous êtes initiés aux limitations imposées par Salesforce, passons à la logique de notre trigger.

Voilà la logique que je propose pour développer notre script

1. Parcours de la liste des cases pour récupérer les mails de ceux qui ne sont pas reliés à un contact (champ « contactId » null) et qui ont le champ « SuppliedEmail » (celui qui nous intéresse) renseigné ;
2. Nous interrogeons Salesforce pour récupérer les contacts qui ont le champ « Email » correspondant aux adresses mails récupérées dans les cases ;
3. Nous re-parcourons la liste des cases qui ne sont pas liés à un contact mais qui ont le champ « SuppliedEmail » renseigné ;
4. Si le contact correspondant à l'adresse mail du case existe parmi ceux qui ont été récupérés dans Salesforce, nous relions le case au contact. Dans le cas contraire, nous créons le contact en lui ajoutant l'adresse mail dans le champ « Email » et également dans le champ « Name » puisque celui-ci est obligatoire pour créer le contact ;
5. Nous créons les contacts dans Salesforce (pour qu'ils obtiennent un identifiant) ;
6. Nous parcourons les cases qui, précédemment, n'avaient pas pu être liés à un contact ;
7. Nous lions chaque case au contact correspondant à l'adresse mail.

Maintenant que je vous ai expliqué la logique du trigger, voici son code :

```
1  trigger JoinContactToCase on Case (
       before insert, before update) {
2
3      public map<String, Contact>
           contactsMap = new map<String,
           Contact>();
4      public map<String, Contact>
           createdContacts;
5      public List<String> mailsList = new
           List<String>();
6      public List<Contact> contactToCreate
            = new List<Contact>();
7      public List<Integer> remainderCases
           = new List<Integer>();
8      Integer caseNumber = 0;
9
10     // Pour chaque case
11     for(Case myCase : Trigger.new){
12
13         // Si le contactId est null mais
                que le champ suppleidMail
               ne l'est pas
14         if(myCase.ContactId == null &&
               myCase.SuppliedEmail != null
               ){
15
16             // Récupération de l'adresse
                    mail
17             mailsList.add(myCase.
                   SuppliedEmail);
18         }
19     }
20
21     // Récupération des contacts associé
           s aux adresses mails
22     contactsMap = getContacts(mailsList)
           ;
23
24     // Parcours des cases
25     for(Case myCase : Trigger.new){
26
27         if(myCase.ContactId == null &&
               myCase.SuppliedEmail != null
               ){
28
29             // Si le contact avec l'
                   adresse mail du case
                   existe dans la map
30             if(contactsMap.containsKey(
                   myCase.SuppliedEmail)){
31                 myCase.ContactId =
                       contactsMap.get(
```

```
                    myCase.SuppliedEmail
                    ).Id;
32              }
33              // On ajoute le contact à la
                    liste pour le créer par
                    la suite
34              else{
35                  contactToCreate.add(new
                        Contact(
36                      Email = myCase.
                            SuppliedEmail,
37                      LastName = myCase.
                            SuppliedEmail
38                  ));
39
40                  // On indique quel case
                        il reste à relier à
                        un contact
41                  remainderCases.add(
                        caseNumber);
42              }
43          }
44
45          caseNumber++;
46      }
47
48      // Création des contacts
49      insert contactToCreate;
50
51      // Récupération des contacts créés
            sous formes de map<email,
            contact>
52      createdContacts = getContactsByMap(
            contactToCreate);
53
54      // Pour chaque case qu'il reste à
            relier à un contact
55      for(Integer theCaseNumber :
            remainderCases){
56
57          // Si le contact de l'adresse
                mail du case est un contact
                venant d'être créé
58          if(createdContacts.containsKey(
                Trigger.new[theCaseNumber].
                SuppliedEmail)){
59              // Liaison du case avec le
                        contact
60              Trigger.new[theCaseNumber].
                    ContactId =
                    createdContacts.get(
                    Trigger.new[
                    theCaseNumber].
                    SuppliedEmail).Id;
61          }
62      }
63
64      /** Retourne les contacts associés
            aux adresses mails données en
            paramètre sous forme de map<mail
            , contact> **/
65      public map<String, Contact>
            getContacts(List<String> mails){
66
67          return getContactsByMap([
68              SELECT Id, Email
69              FROM Contact
70              WHERE Email IN :mails
71          ]);
72      }
73
74      /** Retourne les contacts de la
            liste sous forme de map<mail,
            contact> **/
75      public map<String, Contact>
            getContactsByMap(List<Contact>
            contactsList){
76          map<String, Contact> result =
                new map<String, Contact>();
77
78          // Insère des contacts dans la
                map avec, en clé, l'adresse
                mail
79          for(Contact myContact :
                contactsList){
80              result.put(myContact.Email,
                    myContact);
81          }
82
83          return result;
84      }
85  }
86
```

4 Test du trigger

Lors de toute mise en production, il est obligatoire que le code Apex ait une couverture de code d'au moins 75 % grâce à des tests unitaires sans erreur.

Je vous conseille tout de même de vous rapprocher le plus possible des 100 % de couverture afin de vous assurer qu'il y ait le moins de chances possibles de rencontrer des bogues.

Pour effectuer des tests unitaires, nous allons devoir reproduire l'environnement complet qui va déclencher notre trigger. Il nous faudra donc :

- une liste de cases ;
- une liste de contacts (pour couvrir le cas où un case détient une adresse mail correspondant à un contact déjà existant dans Salesforce).

Nous allons devoir tester deux cas :

- l'adresse mail du case correspond à un contact existant ;
- l'adresse mail du case ne correspond pas à un contact existant.

Voici la classe de tests que j'ai développée pour couvrir le trigger :

```
1   @isTest(seeAllData=false)
2   private class JoinContactToCaseTest {
3
4       public static List<Contact>
            contactsList;
5       public static List<Case> casesList;
6       public static String email1;
7
8       static void init(){
9           contactsList = new List<Contact>
                ();
10          casesList = new List<Case>();
11
```

```
12          email1 = 'toto@toto.com';
13
14          casesList.add(new Case(
15              SuppliedEMail = email1
16          ));
17      }
18
19      /** Test avec un contact existant **
            /
20      static testMethod void
            testWithExistingContact() {
21          init();
22          Test.startTest();
23
24          contactsList.add(new Contact(
25              Email = email1,
26              LastName = email1
27          ));
28          insert contactsList;
29
30          insert casesList;
31
32          // Récupération du case pour ê
                tre sûr d'avoir le champ '
                ContactId' non null
33          casesList = [
34              SELECT Id, ContactId
35              FROM Case
36              WHERE Id = :casesList[0].Id
37          ];
38
39          // Vérification que le case est
                relié au bon contact
40          System.assertEquals(casesList[0
                ].ContactId, contactsList[0
                ].Id);
41
42          Test.stopTest();
43      }
44
45      /** Test avec un contact inexistant
            **/
46      static testMethod void
            testWithDoesntExistingContact()
            {
47          init();
48          Test.startTest();
49
50          insert casesList;
51
52          // Récupération du case pour ê
                tre sûr d'avoir le champ '
                ContactId' non null
53          casesList = [
54              SELECT Id, ContactId
55              FROM Case
56              WHERE Id = :casesList[0].Id
57          ];
58
59          // Récupération du contact créé
                avec le trigger
60          contactsList = [
61              SELECT Id, Email
62              FROM Contact
63              WHERE Email = :email1
64          ];
65
66          // Vérification que le case est
                relié au bon contact
67          System.assertEquals(casesList[0
                ].ContactId, contactsList[0
                ].Id);
68
69          Test.stopTest();
70      }
71  }
```

La première ligne peut vous amener à vous poser des questions. Elle signifie qu'il s'agit d'une classe de tests mais que nous n'utiliserons aucune vraie donnée de Salesforce (via la valeur false du paramètre seeAllData). Vous n'êtes pas obligé d'ajouter la parenthèse avec son contenu, c'est-à-dire que l'annotation @isTest suffit, mais dans cette situation, cela revient à notre cas présent qui précise de ne pas utiliser de vraies données. En revanche, vous pouvez être, dans certaines situations, obligé d'en utiliser de vraies, notamment lorsque vous avez besoin de manipuler des profils, des rôles, des recordTypes, etc. dans votre code. À ce moment-là, vous devrez initialiser ce paramètre à true.

Vous remarquerez que je précise que ma classe est privée, mais ce n'est qu'un détail sans importance, elle aurait pu être publique. Une autre chose à noter est que vous pouvez déclarer votre classe avec with(out) sharing class qui signifie que vous souhaitez utiliser ou non les règles de partage mises en place à l'intérieur de votre organisation. Par défaut, comme dans ce tutoriel, je ne le mets pas et cela équivaut à les utiliser donc, comme si je l'avais déclaré comme private with sharing class.

Allons maintenant un peu plus en profondeur dans le code, je déclare trois variables :

- une liste de contacts ;
- une liste de cases ;
- une adresse mail.

Je pense qu'au vu du scénario, vous devez comprendre leur utilité.

4.1 Méthode init

```
1   static void init(){
2       contactsList = new List<Contact>();
3       casesList = new List<Case>();
4
5       email1 = 'toto@toto.com';
6
7       casesList.add(new Case(
8           SuppliedEMail = email1
9       ));
10  }
```

Afin de factoriser mon code pour éviter la redondance, j'ai pris pour habitude de créer une méthode init() qui me sert à initialiser, comme son nom l'indique, mon environnement de tests, c'est-à-dire les données. Je l'appelle en début de chaque méthode de tests. Ici, elle crée un case mais ne l'ajoute pas dans Salesforce.

Passons maintenant à nos deux tests unitaires qui, je le rappelle, auront pour objectif de tester le trigger dans le cas où un contact aurait la même

adresse mail que le champ suppliedEmail du case et un autre cas où ce contact n'existerait pas.

4.2 Premier cas de test

```
1  /** Test avec un contact existant **/
2  static testMethod void
       testWithExistingContact() {
3      init();
4      Test.startTest();
5
6      contactsList.add(new Contact(
7          Email = email1,
8          LastName = email1
9      ));
10     insert contactsList;
11
12     insert casesList;
13
14     // Récupération du case pour être sû
           r d'avoir le champ 'ContactId'
           non null
15     casesList = [
16         SELECT Id, ContactId
17         FROM Case
18         WHERE Id = :casesList[0].Id
19     ];
20
21     // Vérification que le case est reli
           é au bon contact
22     System.assertEquals(casesList[0].
           ContactId, contactsList[0].Id);
23
24     Test.stopTest();
25 }
```

Il nous faut, dans un premier temps, initialiser notre environnement de tests. Nous appelons pour cela la méthode init.

Nous avons ensuite recours à la méthode statique Test.startTest, mais également à Test.stopTest à la fin de notre test. Nous sommes obligés de faire cela, afin de spécifier que ce qui sera exécuté entre ces deux méthodes, ce seront des tests qui seront soumis aux limitations imposées par Salesforce (ce que je vous ai mentionné légèrement plus haut dans un chapitre précédent, nombre de requêtes SOQL, exécution du temps du script, etc.). Donc, toutes les initialisations de variables et d'environnements doivent se faire en amont, ce que j'ai fait en appelant la méthode init précédemment.

Je demande ensuite à Salesforce de créer un contact puis le case, ce qui aura pour conséquence de déclencher le trigger développé plus haut.

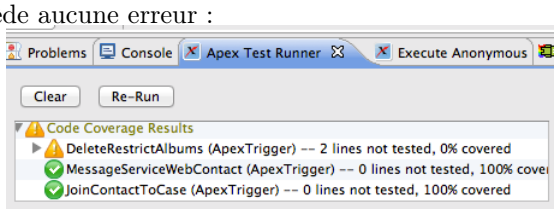
Le case créé doit être rattaché avec notre contact. Pour s'en assurer, nous récupérons le case que nous venons d'enregistrer, une fois le trigger exécuté.

Petite anecdote à savoir à ce sujet, avant d'utiliser insert, nous avons un tableau de cases qui ne possèdent pas de champ Id ; mais après cela, c'est le cas puisqu'ils existeront dans Salesforce. C'est pour cette raison que, lorsque j'écris ma requête SOQL, je peux me permettre d'utiliser ce champ, comme il n'est plus null. Et je préfère récupérer le case en question par mesure de sécurité, car il m'est déjà arrivé de me rendre compte que certains champs sont à null alors qu'ils auraient dû être initialisés. Donc, depuis, j'ai pris l'habitude de récupérer mes résultats.

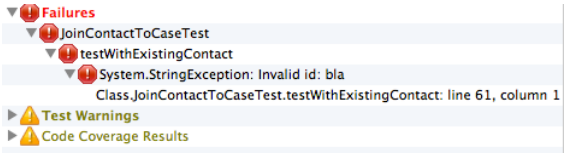
Dernière étape, la vérification des valeurs. Nous utilisons la méthode System.assertEquals en lui transmettant les valeurs que nous voulons vérifier. Dans notre cas, il s'agit donc de l'identifiant du contact, à savoir le champ ContactId du Case et le champ Id du Contact.

Vous n'êtes pas obligé de vous assurer de la cohérence de vos données pour couvrir votre code et mettre en production, mais je vous rappelle que vous devez au minimum avoir une couverture de code de 75 % et sans erreur, et je pense que vous comprendrez qu'il est toujours plus sûr de vérifier que le script fait bien ce que l'on attend de lui.

Voici ce que vous devriez obtenir dans le cas où votre couverture de tests dépasse les 75 % et ne possède aucune erreur :



En revanche, voici ce que vous obtiendrez dans le cas où votre test provoque une erreur :



Dans le cas présent, pour obtenir cette erreur, j'ai modifié les valeurs qui sont vérifiées en spécifiant « bla » comme identifiant du contact du case, ce qui n'est donc absolument pas cohérent.

À noter que le trigger que nous testons s'appelle JoinContactToCase.

4.3 Second cas de test

```
1  /** Test avec un contact inexistant **/
2  static testMethod void
       testWithDoesntExistingContact() {
3      init();
4      Test.startTest();
5
6      insert casesList;
7
8      // Récupération du case pour être sû
           r d'avoir le champ 'ContactId'
           non null
9      casesList = [
10         SELECT Id, ContactId
11         FROM Case
12         WHERE Id = :casesList[0].Id
13     ];
14
15     // Récupération du contact créé par
           le trigger
16     contactsList = [
17         SELECT Id, Email
```

```
18          FROM Contact
19          WHERE Email = :email1
20      ];
21
22      // Vérification que le case est relié au bon contact
23      System.assertEquals(casesList[0].ContactId, contactsList[0].Id);
24
25      Test.stopTest();
26  }
```

Notre second test permet de nous assurer que, si le champ SuppliedEmail du case ne correspond à aucun contact, le trigger le crée pour ensuite le rattacher au case approprié.

La méthode est assez similaire à la première, sauf que nous ne créons pas de contact. Nous insérons seulement notre case pour ensuite le récupérer, lui et le contact créé à l'occasion dans Salesforce.

La finalité est toujours la même, nous vérifions que le case est relié au bon contact.

Il existe de nombreuses méthodes pour vérifier ou non des valeurs ou exécuter un test sous certaines conditions. Nous pouvons, par exemple, nous assurer que deux valeurs sont différentes grâce à la méthode System.assertNotEquals, ou lancer l'exécution en tant qu'un utilisateur donné ou encore vérifier le message renvoyé par une exception à l'aide d'un try ? catch. Si vous êtes curieux et que vous voulez en apprendre plus sur les tests unitaires de Salesforce, je vous invite à lire la documentation et je vous propose quelques liens plus bas dans la partie « Ressources utiles ».

5 Test d'un service web

J'ai eu la chance (ou la poisse, je ne me suis pas encore décidé) de travailler sur un projet où il y a eu l'utilisation d'un service web pour y récupérer des données. Et donc, qui dit service web, dit obligatoirement tests du service web. Nous allons pour cela utiliser ce que nous appelons des données mocks, ce qui signifie que nous n'allons pas tester le service web en lui-même, afin de ne pas dépendre de lui dans le cas où il ne répondrait pas ou ne renverrait pas ce que nous attendons, mais plutôt créer de fausses données qu'il aurait retournées. Je vous invite à lire mon précédent article qui traite sur la façon d'appeler un service web externe à partir de Salesforce : Appeler un webservice externe depuis Salesforce (lien 15).

À propos de la méthode employée, je vais utiliser le fichier WSDL fourni dans l'article précédemment cité pour l'importer dans Salesforce, afin qu'il me génère le client. Et c'est ce client, finalement, que nous allons tester.

Deux méthodes s'offrent à nous que je vais détailler par la suite :

- transmettre une ressource statique à Salesforce contenant des données retournées par le service web ;
- créer nous-mêmes les données renvoyées par le service web et manipulées par le client Salesforce (WSDL2Apex).

5.1 Tester avec une réponse mock

Effectuer des tests unitaires en utilisant des données mocks évite de dépendre de ce que l'on devrait recevoir. Ainsi, dans notre cas, si le web service ne répond pas ou que les données sont différentes de ce qui est attendu, cela n'affectera pas notre test.

Nous allons ici recomposer les données que nous aurait renvoyées le service web une fois parsées par le client Salesforce WSDL2Apex.

Afin de connaître la forme des données que vous devez reconstituer, je vous invite à jeter un ?il aux classes qui ont été générées par Salesforce lorsque vous avez importé votre WSDL.

L'objet que vous utiliserez sera de type Result. À vous de voir ce qu'il contient, vu que cela dépend de ce que vous renvoie le service web (une simple donnée ou une architecture de données). Celui utilisé pour ce tutoriel ne retourne seulement que la valeur de l'opération mathématique, soit un nombre de type Double. Reconstruire les données est donc ici assez simple.

Il n'est pas difficile d'implémenter une réponse mock. Pour cela, vous devez créer une classe qui implémente l'interface WebServiceMock et y développer une méthode doInvoke qui dispose de plusieurs paramètres, puis utiliser l'un d'entre eux (response) pour y stocker votre donnée mocks.

Voici la classe utilisée pour le service web :

```
1  @isTest
2  global class
       CalculatorWebServiceMockDivide
       implements WebServiceMock {
3     global void doInvoke(
4              Object stub,
5              Object request,
6              Map<String, Object> response,
7              String endpoint,
8              String soapAction,
9              String requestName,
10             String responseNS,
11             String responseName,
12             String responseType) {
13
14         calculatorWebService.
               divideResponse_element
               respElement = new
               calculatorWebService.
               divideResponse_element();
15         respElement.Result = 5.0;
16         response.put('response_x',
               respElement);
```

```
17      }
18  }
```

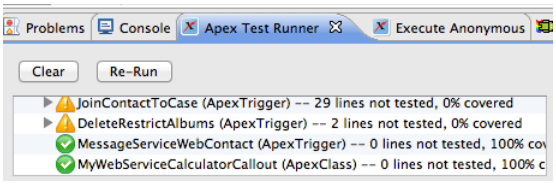
Nous commençons par déclarer notre classe comme une classe de tests (annotation @isTest), puis nous créons un objet du type correspondant à la classe de réponse de la méthode du service web que nous appelons. Nous reconstituons ensuite les données qu'est censée retourner la méthode appelée via l'attribut Result de la classe. Puis nous finissons par attribuer notre objet de type Result à la map réponse au niveau de la clé response_x.

C'est tout ce qu'il faut effectuer pour créer une réponse mock dans Salesforce. Passons maintenant à notre classe de tests unitaires pour apprendre comment l'utiliser :

```
1  static testMethod void testWithMock() {
2      Test.startTest();
3
4      Double numerator = 10.7;
5      Double denominator = 12.9;
6
7      // Remplace la réponse du
           webservice par la réponse mock
8      Test.setMock(WebServiceMock.class,
           new
           CalculatorWebServiceMockDivide
           ());
9
10     MyWebServiceCalculatorCallout
           myWebServiceCalculatorCallout =
            new
           MyWebServiceCalculatorCallout()
           ;
11
12     // Compare des valeurs
13     System.assertEquals(5.0,
           myWebServiceCalculatorCallout.
           divide(numerator, denominator))
           ;
14
15     Test.stopTest();
16 }
```

La méthode est assez semblable à ce que j'ai pu vous présenter au-dessus. Nous avons les méthodes Test.startTest et Test.stopTest en début et fin de la méthode pour délimiter la zone de test, mais ce qui change, c'est l'appel à la méthode Test.setMock en donnant en premier paramètre l'interface WebServiceMock (celle que notre service mock implémente), les paramètres que nous devons transmettre à la méthode d'appel du web service, ainsi qu'une instance de notre classe de données mocks. Cela signifie qu'à partir de ce moment-là, chaque appel au service web ne retournera uniquement que les données mocks que nous avons transmises à notre environnement. Nous terminons comme d'habitude par nous assurer de la valeur retournée, qui est 5.0 pour le test effectué.

Une fois le test passé, nous pouvons vérifier que tout s'est bien déroulé grâce aux logs. Voyez par vous-même (classe MyWebServiceCalculatorCallout) :



5.2 Tester avec une ressource statique

Il existe un deuxième moyen pour tester un service web, qui consiste à utiliser une ressource statique qui détient le contenu retourné par l'appel à une méthode du web service. Je ne pense pas qu'il soit nécessaire que je vous explique comment uploader une ressource statique dans Salesforce, j'estime qu'il s'agit d'un prérequis. Vous pouvez récupérer ce fichier dans une archive contenant toutes les ressources que je vous mets à disposition à la fin de ce tutoriel dans la partie Conclusion.

```
1  /** Test avec une ressource statique **/
2  static testMethod void
       testWithStaticResource(){
3      Test.startTest();
4
5      Double numerator = 10.7;
6      Double denominator = 12.9;
7
8      // Crée la réponse mock
9      StaticResourceCalloutMock mock = new
           StaticResourceCalloutMock();
10     // Utilise la ressource statique
           divideResult
11     mock.setStaticResource('divideResult
           ');
12
13     // Attribution de la réponse mock
14     Test.setMock(HttpCalloutMock.class,
           mock);
15
16     MyWebServiceCalculatorCallout
           myWebServiceCalculatorCallout =
           new
           MyWebServiceCalculatorCallout();
17
18     // Comparaison des valeurs
19     System.assertEquals(5.0,
           myWebServiceCalculatorCallout.
           divide(numerator, denominator));
20
21     Test.stopTest();
22 }
```

Je ne vais pas vous refaire la même description que d'habitude. Ce qui diffère ici, c'est que vous devez créer un objet de type StaticResourceCalloutMock et lui transmettre le nom de la ressource statique que vous voulez utiliser pour les tests, pour finir par faire utiliser la méthode Test.setMock en donnant, en premier paramètre, la classe HttpCalloutMock, ainsi que le mock en question en second paramètre.

6 Conclusion

C'est ici que se termine ce tutoriel. J'espère vous avoir initié au minimum de ce que vous devez savoir pour effectuer des tests unitaires dans vos développements d'applications Salesforce.

Certes, les cas présentés ici sont relativement simples à couvrir et à tester, mais l'objectif est surtout de vous montrer comment procéder.

Choses promises, choses dues, je vous partage une archive contenant les ressources utilisées tout au long de ce tutoriel, que vous pouvez télécharger ici (lien 16 (ou consulter directement les sources sur Github (lien 17) ou Bitbucket (lien 18)), l'ensemble contient :

- le trigger (JoinContactToCase) ;
- la classe de tests du trigger (JoinContactToCaseTest) ;
- le WSDL du service web (calculator.xml) ;
- la réponse mock du service web (CalculatorWebServiceMockDivide) ;
- la ressource statique représentant une réponse du service web (divideResult.xml) ;
- la classe de tests du service web (CalculatorWebServiceTest) ;
- la classe de résultats des données du service web (CalculatorWebService).

7 Ressources utiles

Si vous souhaitez obtenir plus d'informations sur la couverture de code, je vous invite à visiter ces liens :

- Une introduction méthodes de tests Apex : lien 19 ;
- Tester son code avec des cas positifs et négatifs : lien 20 ;
- Tester l'appel d'un service web avec une réponse mock : lien 21 ;
- Tester l'appel d'un service web avec une ressource statique : lien 22 ;
- Vérifier sa couverture de code : lien 23 ;
- Meilleures pratiques de tests : lien 24.

Retrouvez l'article de ***Aurélien Laval*** *en ligne : lien 25*

2D/3D/Jeux

Les dernières news 2D/3D/Jeux

La vérité sur la qualité des pilotes graphiques OpenGL

Rich Geldreich, développeur chez Valve, écrit sur un blog ses opinions personnelles (donc, à ne pas lier avec Valve) sur OpenGL. Son opinion est intéressante, notamment car Rich a été développeur sur le premier moteur utilisant la technique de rendu différé (pour Shrek, sur Xbox). Ensuite il a aussi créé une bibliothèque de compression avancée pour le DXTc, pour Ensemble Studios, et travaille maintenant chez Valve, notamment sur le portage et l'optimisation des jeux Portal 2 et DoTA 2 sur Linux. Il est aussi un des développeurs de vogl, un nouveau débogueur/profileur OpenGL multiplateforme.

Dans son blog, il revient donc sur les différents supports OpenGL disponibles sur le marché. Voici ce qu'il en dit :

1 Vendor 1

C'est l'implémentation la plus utilisée chez les développeurs, car c'est l'implémentation la plus avancée du marché. C'est le pilote « standard », il est très rapide et ses développeurs préfèrent la sécurité plutôt que de suivre la spécification à 100 %. Les développeurs jouant avec OpenGL utilisent ce pilote car il possède les extensions les plus amusantes. La plupart des choses que vous entendez pour concurrencer Mantle/Direct3D 12 sont réalisées par les développeurs jouant avec ce pilote. Le problème, c'est qu'il n'est pas possible de cibler uniquement celui-ci, sans quoi, une grande partie du marché serait oubliée.

Toutefois, lorsque le moteur Source 1 a été porté sur Linux et que les développeurs de Valve tenaient la main aux développeurs de ce pilote, ils n'étaient même pas capables de mettre à jour un tampon (avec glMap() (lien 26) ou glBufferSubData() (lien 27), comme on le ferait en Direct3D 9/11) sans que le pilote ne se bloque. Ici, on parle des choses de « driver perf 101 ». De plus, lorsque vous tombez sur un bogue dans ce pilote, celui-ci s'étale complètement la figure contre le sol et crashe soit le GPU, soit Windows. Malgré tout, c'est un pilote solide et sûr.

Cette implémentation supporte énormément d'extensions (dont certaines à la pointe de la modernité) fonctionnant plus ou moins bien, mais dès que vous utilisez les plus importantes, vous sortez des clous de sécurité du pilote et vous tombez dans un terrain miné.

Historiquement, les outils de ce développeur sont nuls ou ne fonctionnent que pendant un certain temps puis arrêtent de fonctionner ou ne fonctionnent que si vous suppliez l'aide auprès de l'équipe de ces outils. Ils possèdent des outils, peut-être comme ceux dans la série Dilbert leur permettant de savoir ce qui se passe. Bien sûr, ces outils ne fonctionnent que sur leurs pilotes.

Ce Vendor est très précautionneux et stratégique lors de l'intégration de leurs développeurs dans les jeux clés afin que les choses marchent. C'est une épée à double tranchant, car ces développeurs vont refuser de déboguer les problèmes des autres développeurs de pilotes et leur vision d'OpenGL ne se fait qu'au travers de leur implémentation. Les développeurs intégrant une équipe feront les choses pour leur pilote, sans se soucier de l'impact sur les autres implémentations.

Historiquement, ce développeur fera des choses comme remplacer la globalité des shaders des jeux clés par leur implémentation afin que cela fonctionne mieux sur leur implémentation (et cela le fait effectivement très bien). La plupart des pilotes font certainement cela occasionnellement, mais ce développeur fera tout pour la performance. Qu'est-ce que cela signifie pour l'industrie du jeu vidéo ? C'est que vous,

développeur d'effets et de graphismes, vous n'atteindrez jamais le même niveau technique dans votre jeu (même si vous utilisez les mêmes algorithmes !), car vous n'avez pas les ingénieurs du pilote travaillant spécialement sur votre jeu permettant au pilote de faire exactement ce qu'il doit (en utilisant des shaders bas niveau optimisés) lorsque votre moteur fonctionne. Cela veut aussi dire que les légendes du monde graphique que vous connaissez ne sont pas aussi intelligentes ou capables que ce que l'histoire montre : elles ont reçu beaucoup d'aide.

Pour plaisanter, ce développeur est connu sous le nom de « mafia graphique ». Soyez prudents, si leurs développeurs débarquent dans votre équipe... ils sont sérieux.

2 Vendor 2

Le pilote est complètement désordonné, possède des performances inconsistantes, est très bogué et les tests de régression sont incomplets, même le parallélisme du pilote est hors contrôle des développeurs officiels. Malheureusement, les GPU de ce constructeur sont standards et sont très corrects, donc vous ne pouvez pas les ignorer. Les fonctions basiques comme glTexStorage() (lien 28) crashent (sur un titre publié) et cela, depuis des mois. Toutefois, les développeurs respectent mieux la spécification que les développeurs du Vendor 1, mais cela n'est pas une bonne chose car la plupart des développeurs utilisent le matériel du premier Vendor et lorsque cela ne marche pas sur ce second matériel, c'est ce Vendor qui est critiqué et non l'implémentation d'OpenGL de l'autre.

Les extensions clés ne fonctionnent simplement pas. Elles sont là, simplement pour montrer l'avancement aux managers. La plupart des développeurs OpenGL n'utilisent jamais ces extensions car elles ne fonctionnent pas. Elles semblent géniales sur le papier et indiquent la progression mais finalement, c'est une démonstration parfaite du non fonctionnement des extensions OpenGL dans la pratique.

Ce pilote ne peut avoir de choses comme les requêtes ou les synchronisations de fonctionnel. Donc n'importe quelle extension reposant sur les synchronisations CPU/GPU ne peuvent pas fonctionner. Les développeurs restant chez ce constructeur souhaitent tout simplement travailler chez le premier Vendor.

Cette entreprise ne peut mettre à jour son pilote sans rien casser. Ils vont vous envoyer des mises à jour ou des correctifs pour corriger un bogue et en rajouter deux. Si vous effectuez du pas à pas sur les points d'entrées de ce pilote, vous allez remarquer des couches et des couches de croûte clouées au fil des ans provenant de développeurs ayant quitté l'entreprise. Personne ne comprend ces couches pour pouvoir les modifier sans risque.

Rich a quelques fois vu des choses bizarres se produisant dans ce pilote lorsqu'il rejouait des flux d'appels OpenGL de jeux commerciaux avec vogl-replay. Le jeu en lui-même fonctionne correctement, mais lorsque le flux d'appels d'OpenGL est rejoué, on peut voir une corruption massive du tampon d'image (qui s'en va après sur chaque flush du pipeline OpenGL). Il suppose que le pilote désactive des fonctionnalités entières trop boguées suivant des profils d'applications.

Néanmoins, ce développeur possède une petite équipe pour les outils qui a créé des outils de débogage pratiques et qui fonctionnent la plupart du temps, tant que vous utilisez leurs GPU. Sans ses outils, le portage du moteur Source 1 aurait pris bien plus de temps.

Cela aurait pu être temporaire, mais il semble que du point de vue de la fiabilité cela ne fait qu'empirer (et oui, cela peut être encore pire).

Toutefois, ils connaissent la spécificationÕpenGL complètement et cela à la syllabe prêt. Si vous pouvez obtenir leur assistance, leurs conseils sont plus ou moins raisonnables pour la spécification d'OpenGL (pas les extensions).

3 Vendor 3 - Pilote 1

C'est compliqué d'être sincèrement en colère contre ce troisième développeur. Ils ne veulent pas faire de graphismes. Ils ne le font que par distraction, la mode étant de tout intégrer sur une même puce et vu qu'ils ont énormément d'espace à partager sur celle-ci, ils le font. Ce sont des maîtres au niveau matériel, mais du côté logiciel, ils ne sont pas vraiment intéressés. Ils sont les leaders au niveau des pilotes graphiques open source et leurs spécifications matérielles sont presque publiques. Ils ont actuellement tellement d'argent et leur organisation est si grande qu'ils peuvent se permettre d'avoir deux équipes différentes de développeurs pour le pilote ! Hé oui, pour ce développeur, vous pouvez avoir un premier pilote OpenGL pour une plateforme et un second sur une autre. Ce sont deux équipes différentes avec deux codes différents.

En tout cas, l'équipe des ressources humaines de ce développeur est intelligente : elle engage directement les maîtres open source pour faire avancer la-

borieusement le premier pilote. Ce pilote est moins avancé que les pilotes principaux, mais il fonctionne plus ou moins, tant que vous ne comprenez pas ou que vous n'êtes pas intéressé par le terme « FPS ». Si vraiment il ne fonctionne pas et que vous êtes motivé, vous pouvez essayer de corriger vous-même le pilote et soumettre un patch. Si vous êtes vraiment bon à ce petit jeu, vous pouvez même obtenir un emploi chez ce développeur.

Malheureusement, ce premier pilote est très en retard dans la spécification OpenGL, mais peut-être que dans une année ou deux, ils rattraperont ce retard et implémenteront la spécification de l'année dernière. Mais vous ne pouvez pas ignorer ce pilote car ils ont un marché important et en expansion. Donc, si un développeur de jeux souhaite atteindre ce marché, il ne peut pas utiliser les extensions amusantes ou les dernières améliorations d'OpenGL « modernes » supportées par le Vendor 1 et 2.

Ce Vendor ne propose absolument aucun outil pour OpenGL. Si vous souhaitez déboguer votre programme... bienvenue en 1999.

4 Vendor 3 - Pilote 2

Un désastre complet. Le pilote de cette seconde équipe est à peine utilisé par les titres OpenGL. Il y a tellement de morceaux de code qui ne fonctionnent pas. Ils ne peuvent pas mettre à jour un tampon sans une corruption massive aléatoire. Cette équipe va faire des choses comme vous donner un pilote bogué unique et différent pour chaque titre afin de réaliser des analyses de performances et des tests. Cette équipe vous demande honnêtement si c'est la performance ou la justesse qui est le plus important.

Rich a vu une équipe de développeurs bien connue passer plus d'une année pour avoir leur moteur de rendu OpenGL 4.X avec extensions fonctionnel sur ce pilote. Ce pilote ne fonctionne simplement pas. Faites simplement un moteur de rendu OpenGL 3.X avec des hacks.

Du bon côté, le Vendor 3 fournit plus d'informations internes sur leur matériel que la première équipe, faisant que le pilote a tendance à être un petit peu plus rapide, lorsqu'il fonctionne.

5 Les autres pilotes

En plus des implémentations précédentes, il existe aussi des pilotes open source, développés par la communauté pour les puces du Vendor 1 et 2. Ils sont souvent en retard d'un point de vue OpenGL, mais il parait que cela fonctionne bien. Rich n'a pas d'expérience réelle avec ces pilotes, car il a toujours été craintif de travailler avec ces pilotes open source/développés à l'aide de rétro-ingénierie ce qui aurait pu énerver les équipes de développement des pilotes fermés.

Le Vendor 1 déteste ces pilotes car ils sont profondément établis dans la façon dont les choses sont faites. Ces développeurs reçoivent des fonds et hypothèques pour subsister et il y a donc beaucoup d'inertie. Il n'y a absolument aucune chance qu'ils publient leurs spécifications top secrètes ou même qu'ils rendent open source leur pilote. Le Vendor 1 devra sauter dans le train de l'open source prochainement afin de mieux concurrencer le modèle ouvert du troisième Vendor, que cela leur plaise ou pas.

Le deuxième Vendor aide sans enthousiasme le pilote open source en fournissant une petite équipe les aidant à maintenir les choses fonctionnelles. À un certain point, le pilote open source pour le deuxième Vendor pourra être plus viable que leur pilote fermé semi-fonctionnel.

6 Conclusion

Pour publier votre jeu, vous devriez tester votre code sur chaque pilote et contourner tous les problèmes. Que les « dieux d'OpenGL » vous aident si vous faites face à des corruptions de GPU aléatoires, des corruptions de pile, des freezes ou des crashs. Soyez très gentils avec les équipes de développement des pilotes et leurs managers, car sans eux, vos chances sont quasi nulles.

*Commentez la news d'**Alexandre Laurent** en ligne : lien 29*

XNA est-il mort ? Doit-on encore s'intéresser à la technologie de Microsoft pour faire des jeux ?

1 Présentation

En tant que développeur de jeux vidéo, vous avez sûrement entendu parler de XNA. Peut-être même que vous vous y êtes intéressé et que vous avez développé des programmes ou des jeux avec. Pour rappel, XNA est une série d'outils proposée par Microsoft pour faciliter le développement de jeux pour les plateformes Windows, Zune, Windows Phone 7 et même la Xbox 360. Pour cela, XNA repose sur le langage C#, la bibliothèque de jeux DirectX, l'éditeur Visual Studio, le débogueur GPU PIX, l'outil audio XACT. Ce framework a été une des premières opportunités offerte aux développeurs indépendants de publier des jeux sur une console (la Xbox 360) et d'avoir une visibilité augmentée à travers le Xbox Live. Le public visé par cet ensemble d'outils sont les étudiants, les passionnés et les développeurs indépendants. Afin de bénéficier de la publication sur le Xbox Live Marketplace, il est nécessaire d'avoir un compte premium vendu au prix de 99 $ par an. Vous pouvez découvrir une présentation approfondie (analyse de l'architecture, description technique...) dans cet article : lien 30.

2 Historique

La toute première version de XNA (appelée XNA Game Studio Express) a été publiée le 11 décembre 2006. Durant l'année 2007, une seconde version (XNA Game Studio Express 1.0 Refresh) est apparue, proposant une mise à jour de la version précédemment nommée. Ensuite, quatre autres versions ont été distribuées, toutes nommées XNA Game Studio (2.0, 3.0, 3.1, 4.0). La dernière version (4.0) a été publiée le 9 mars 2010. Pour information, la version 3.0 apportait le support de Zune et de la communauté Xbox Live, le C# 3.0, LINQ et la plupart des versions de Visual Studio 2008. La version 3.1 rajoutait le support du playback vidéo, une nouvelle version de la bibliothèque audio et le support des avatars Xbox 360. Quant à la dernière version, elle apportait le support de Windows Phone 7.5 et de Visual Basic.

Avec cette version estampillée 4.0, les MVP (Most Valuable Professionals) ont été avertis que le support s'arrêterait courant avril 2014. Aucune autre version du framework n'est prévue et Microsoft ne travaille plus du tout dessus, comme il est possible de l'apprendre dans cette actualité : lien 31. De plus, lors de la sortie de la quatrième version, des utilisateurs étaient confus par le rapprochement du framework avec le SDK de Windows Phone et à l'époque déjà, certains présageaient la mort du framework.

3 Les jeux développés avec XNA

De nombreux développeurs ont utilisé cette technologie pour publier leurs jeux, que ce soit sur Xbox 360 ou sur PC. Nous pouvons citer Terraria, Fez, Escape Goat, Dust : An Elysian Tail.... Il serait bien dommage d'oublier Chibis Bomba, un jeu présenté dans nos pages.

4 Les alternatives

Comme vous l'avez compris, Microsoft ne compte plus faire de mise à jour. Le support de Windows 8, ou des nouvelles versions de Visual Studio ne sera jamais réalisé. Pourtant, la technologie n'est pas morte. Cela est possible grâce au projet open source MonoGame. MonoGame est une implémentation libre de la bibliothèque XNA 4. La deuxième bonne nouvelle est que l'équipe ne se limite pas au support des plateformes Microsoft, car il vous sera aussi possible de cibler Mac OS, Linux, Android, PlayStation Mobile, OUYA ainsi que Windows 8 et Windows Phone 8. Le projet est toujours en développement actif et la version 3.2 a été publiée le 7 avril 2014. De nombreux projets professionnels l'utilisent déjà comme Bastion ou encore Fez. Il faut aussi noter que Microsoft utilise MonoGame pour supporter Windows 8 dans leur propre projet : Kodu (lien 32).

Si vous cherchez à tout prix à installer XNA 4 sur Visual Studio 2012 ou 2013, vous pouvez utiliser cet outil qui se chargera d'activer le framework dans les éditeurs de Microsoft.

5 Conclusion

Lors de sa sortie, le framework avait donné une opportunité très intéressante pour les étudiants qui souhaitaient s'insérer facilement dans le monde du développement de jeux vidéo, notamment grâce à une bibliothèque simplifiée et l'utilisation du langage C#, mais aussi pour les développeurs indépendants souhaitant cibler la console de jeux de Microsoft. Malheureusement, Microsoft abandonne finalement cette technologie, possiblement car le marché est devenu bien plus compétitif avec des solutions comme Unity. Toutefois, grâce au projet Monogame, les studios reposant sur XNA peuvent continuer leurs travaux et même s'offrir un support de plateforme étendu.

*Commentez la news d'**Alexandre Laurent** en ligne : lien 33*

Les derniers tutoriels et articles

QB64 : Installation et introduction au Quick Basic

Vous souhaitez vous lancer dans la conception d'un jeu vidéo et cela, avec la volonté de programmer et de mettre les mains dans le cambouis. Alors QB64 est fait pour vous et ce tutoriel sera votre guide pour découvrir ce merveilleux moteur et langage de programmation.

1 Introduction

Vous souhaitez faire un jeu vidéo et vous souhaitez programmer ? Vous voulez faire un jeu vidéo rapidement et simplement, sans pour autant passer par un logiciel de création de jeu vidéo ? Vous êtes alors au bon endroit !

1.1 QB64

QB64 est un compilateur (logiciel transformant le code source en exécutable) pour le langage de programmation **BASIC** pour Windows, Linux et Mac OS X. De plus, QB64 propose un éditeur apportant de nombreuses fonctionnalités vous aidant lors de l'écriture de votre code.

Si vous préférez utiliser un autre éditeur de code, cela est aussi possible.

De plus, QB64 permet la programmation 64 bits grâce à l'ajout des types en 64 bits ainsi qu'un meilleur support du son et des graphismes et offre ainsi une solution moderne tout en utilisant le langage simple BASIC.

1.1.a Spécificités

QB64 est un choix intéressant pour un débutant en programmation souhaitant développer un jeu vidéo pour les raisons suivantes :

- un langage de programmation simple : le BASIC (**B**eginners **A**ll-purpose **S**ymbolic **I**nstruction **C**ode) ;
- une compilation du programme en C++, permettant d'atteindre des hautes performances ;
- des améliorations pour les graphismes (reposant sur la SDL) et pour le support audio. Ainsi, vous pouvez créer des fenêtres d'une résolution supérieure à 640 x 480 et ouvrir des fichiers sonores comme les fichiers MP3 et OGG ou les fichiers image comme les fichiers PNG et JPEG. Pour ces raisons, QB64 est un logiciel convenable pour créer un jeu vidéo.

1.1.b Exemples de jeux

Pour vous convaincre, voici quelques jeux réalisés avec QB64 :

1.1.b.a Papi Commando

Papi Commando est un jeu de tir vu de dessus d'un membre de Developpez.com : lien 34.
Voir la vidéo : lien 35

1.1.b.b Black Annex

Un jeu 2D d'infiltration, espionnage et sabotage dans des bureaux d'entreprises.
Voir la vidéo : lien 36

1.1.b.c Barbarian Remake

Barbarian est un jeu oldschool de combat sorti sur les vieux micro-ordinateurs.
Voir la vidéo : lien 37

2 Installation

L'installation de QB64 est très rapide.

2.1 Windows

Sous Windows, il suffit de télécharger le fichier .7z à partir du site qb64.net (lien 38) (prenez la dernière version) et de décompresser le fichier dans le répertoire que vous souhaitez.

2.2 Linux

Sous Linux, vous devez télécharger le fichier .tar.gz à partir du site .gzqb64.net (lien 39) (prenez la dernière version) et de décompresser le fichier dans un répertoire. Ensuite, vous devez exécuter (dans un terminal) le script setup_lnx.sh. Celui-ci vous guidera dans le processus d'installation en vérifiant et

vous indiquant quelles sont les dépendances manquantes. Une fois prêt (et que tout s'est bien passé), QB64 se lance, validant l'installation.

Sous Linux, l'éditeur BASIC de QB64 (version 0980) souffre d'un problème de performance. En effet, même le simple fait de taper du code provoquera des latences insupportables. Le problème est connu mais ne dispose pas de solution au moment de l'écriture de ce tutoriel. Le mieux est de choisir un autre éditeur (de le configurer pour colorer le code source en suivant la syntaxe BASIC) et de compiler en ligne de commande.

2.3 Mac OS X

Sous Mac OS X, vous devez télécharger le fichier macosx.tar.gz à partir du site qb64.net (lien 40) (prenez la dernière version). Ensuite, vous devez installer Xcode à partir du magasin d'applications (QB64 utilise le compilateur installé par Xcode, vous n'aurez pas besoin directement de Xcode). Sur Mac OS X Lion ou supérieur, vous devez installer les outils en ligne de commande (*command line tools*). Vous pouvez le faire à partir de Xcode, par le menu « Préférences » (« Preferences ») -> « Téléchargements » (« Downloads ») -> « Composants » (« Components ») puis en cliquant sur « Installer » (« Install ») sur la ligne correspondant à l'outil. Finalement, vous pouvez extraire le fichier QB64 téléchargé et exécuter le fichier setup_osx.command.

2.4 Compilation d'un programme en ligne de commande

Vous pouvez utiliser votre propre éditeur pour programmer en QB64. Lorsque vous voulez compiler, il suffira d'appeler le programme qb64 avec l'option -c et le nom de votre fichier pour qu'il soit compilé.

3 Langage

Le langage est donc un dérivé du BASIC. Il est simple et intuitif.

3.1 Hello World

Commençons par le classique Hello World. Cela permettra de vérifier que votre configuration des outils fonctionne correctement.

Voici :

```
1  CLS
2  PRINT "Mon premier programme QB64 !"
3  END
```

La première ligne contient juste une commande (CLS) et permet de nettoyer l'écran (Clear Screen). La seconde, permet d'afficher (PRINT) un texte, qui suit directement la commande PRINT. Lorsque l'on fait suivre une commande de quelque chose, on appelle la partie qui suit la commande « un argument ». La dernière ligne indique simplement la fin (END).

Ici, si vous enlevez le END, cela reviendra au même, car QB64 en voyant la fin du programme, se met en pause et attend une touche.

Le programme affichera donc :

```
1  Mon premier programme QB64 !
```

3.1.a Les commandes

Les commandes sont des mots clés qui sont exécutés et qui ne retournent pas de valeur.

Les commandes sont souvent des mots d'anglais comme : FOR, NEXT, IF, THEN, GOTO. Quelquefois, les mots clés peuvent être des abréviations, comme pour CLS (CLear Screen). Une commande peut prendre en paramètres des arguments : des variables qui seront utilisées pendant l'exécution de la commande.

3.1.b Les arguments

Les arguments pour les commandes ne sont généralement pas placés entre parenthèses. Toutefois, cela est possible et notamment, pratique pour la lecture, pour des arguments décrivant par exemple les coordonnées X et Y.

Quelquefois, un ou des arguments peuvent être optionnels. Dans un tel cas, ils recevront une valeur par défaut.

Chaque argument est séparé par une virgule :

```
1  LINE (160, 100)-(170, 110), , B
```

Ici, on peut voir une série d'arguments, dont un, sans valeur, qui utilisera donc la valeur par défaut.

3.1.c Les variables

Les variables sont des éléments qui contiennent des valeurs (un nombre, par exemple, mais aussi une chaîne de caractères (un texte)) et ces valeurs peuvent changer au cours de l'exécution du programme. Les variables peuvent avoir le nom que vous souhaitez, mis à part celui des commandes (sinon le compilateur considérera cela comme une commande et non une variable).

```
money = 1000
PRINT money
```

Ce code affichera « 1000 ».

Si vous souhaitez afficher une variable et que vous voulez la faire précéder d'une chaîne de caractères, vous devez rajouter un ';' pour les séparer :

```
score = 1000
PRINT "Mon score est : " ; score
```

3.1.d Les chaînes de caractères

Les chaînes de caractères sont des variables contenant du texte. Afin de les différencier des autres variables, il est d'usage de rajouter un '$' après le nom de la variable.

Vous pouvez aussi ajouter du texte simplement avec l'opérateur '+' :

```
site$ = "Developpez" + ".com"
```

3.1.e Les entrées utilisateurs

À un certain point dans votre parcours en programmation, vous allez vouloir demander des informations à l'utilisateur, par exemple, pour qu'il définisse la valeur d'une variable. Cela est possible avec la commande INPUT :

```
INPUT "Votre nom : " ; nom$
```

Cela fera que le programme affiche « Votre nom : » et attendra que l'utilisateur entre son nom. La variable « nom$ » prendra la valeur de ce que l'utilisateur a entré.

3.1.f Les commentaires

La programmation n'est pas une tâche toujours aisée. Nous écrivons du code qui est la transformation d'une idée en un langage. Cela peut paraître simple, mais le jour (que ce soit le lendemain, ou dans un mois) où vous revenez sur votre code et que vous essayez de le relire, l'idée que vous aviez en l'écrivant sera peut-être oubliée.

Pour éviter ces problèmes et rendre la lecture d'un code aisée, vous pouvez rajouter des commentaires. Les commentaires sont du texte, qui ne sera pas utilisé par le compilateur. Vous pouvez écrire tout ce que vous voulez, il est pour vous.

En BASIC, un commentaire commence par ' :

```
' Ceci est un commentaire
' Je peux écrire ce que je veux. Even in
    english.
' Je peux aussi mettre du code toto = 5
' Qu'il soit juste ou pas, peu importe,
   le compilateur ignore les
   commentaires
solution = 42 ' Je peux aussi écrire des
    commentaires après un code
' Le but d'un commentaire n'est pas de d
   écrire ce que signifie la ligne,
   mais sa conséquence
' Ici, j'ai simplement voulu définir la
   solution à ma variable, variable qui
    sera utilisée plus loin et comparée
    avec ce que l'utilisateur va ré
   pondre
```

3.2 Déroulement du programme

Le programme suit généralement le code écran, ligne par ligne, de haut en bas. Mais il devient vite nécessaire de briser ce déroulement linéaire, afin de lui faire répéter des choses, ou encore, de n'exécuter des morceaux de code uniquement si certaines conditions sont vérifiées.

3.2.a Les conditions

Les conditions permettent d'indiquer à l'ordinateur d'exécuter un code, uniquement si certains facteurs sont vérifiés. Pour indiquer une telle condition, la commande s'appelle IF (si). Après le IF, vient la condition à vérifier. Et pour indiquer l'action à faire en cas de commande vérifiée, vous devez utiliser THEN (alors), suivi de l'action à exécuter :

```
IF money < 1000 THEN PRINT "Vous ne
   pouvez pas acheter ceci"
```

Qui peut se traduire par :

```
Si le contenu de la variable
```

Parmi les tests, vous pouvez utiliser :

<	Inférieur à
>	Supérieur à
=	Égal à
<>	Différent de
>=	Supérieur ou égal à
<=	Inférieur ou égal à

De plus, il existe deux autres mots clés : AND (ET) et OR (OU) qui permettent de lier des conditions :

```
IF money < 1000 AND diamant < 5 THEN
   PRINT "Vous ne pouvez pas acheter
   ceci"
```

Qui peut se traduire par :

```
Si le contenu de la variable money est
   inférieur à 1000 et que le contenu
   de la variable diamant est inférieur
    à 5, alors, afficher " Vous ne
   pouvez pas acheter ceci ".
```

3.2.b Les boucles

Les boucles permettent de répéter un morceau de code plusieurs fois.

3.2.b.a La boucle FOR

La boucle FOR permet de répéter un code, en utilisant un compteur. À chaque passage dans la boucle, le compteur va être incrémenté d'une valeur (STEP (pas)). Lorsqu'il a atteint une valeur finale (TO), la boucle n'est plus exécutée et le programme continue :

```
1 FOR i = 0 TO 5 STEP 1
2     PRINT "J'ai ecrit " ; i ; " fois
          cette phrase"
3 NEXT
```

Le mot clé NEXT, permet d'indiquer la fin de la boucle.

Qui affichera :

```
1 J'ai ecrit 1 fois cette phrase
2 J'ai ecrit 2 fois cette phrase
3 J'ai ecrit 3 fois cette phrase
4 J'ai ecrit 4 fois cette phrase
5 J'ai ecrit 5 fois cette phrase
```

Après la boucle, la variable i contient 6.

Le code peut se traduire ainsi :

```
1 Pour une variable i, ayant pour valeur
      initiale 0. La boucle s'arrêtera
      lorsque la variable i vaudra 5. La
      variable i sera incrémentée de 1 à
      chaque tour (chaque appel à NEXT).
      Afficher le message " J'ai ecrit '
      contenu de la variable i' fois cette
       phrase ".
```

Le NEXT permet d'effectuer l'incrémentation de la variable compteur (dans l'exemple, le compteur est la variable i). Le NEXT indique aussi que le programme doit revenir à la ligne du FOR et vérifier la valeur finale, avant de réexécuter (ou non) le code du FOR.

Il n'est pas obligatoire d'écrire STEP 1 lorsque l'incrément est de 1. En effet, c'est sa valeur par défaut.

Il est possible d'avoir un STEP négatif. Veuillez toutefois donner une valeur initiale plus grande que la valeur finale, sinon la boucle sera simplement ignorée.

3.2.b.b La boucle DO

La boucle DO vous permet de créer des boucles s'exécutant tant qu'une condition est vraie (ou tant qu'une condition n'est pas vérifiée).

```
1 i = 0
2 DO WHILE i <= 5
3     PRINT "J'ai ecrit " ; i ; " fois
          cette phrase"
4     i=i+1
5 LOOP
```

Cette boucle effectue la même chose que la boucle FOR précédente. Elle peut être traduite de la façon suivante :

```
1 Une variable i ayant pour valeur 0.
2 Faire, tant que le contenu de la
      variable i est inférieur ou égal à 5
       :
3 afficher " J'ai ecrit 'contenu de la
      variable i' fois cette phrase ".
4 Incrémenter i.
5 Boucler (revenir à la condition du DO).
```

À la place du WHILE, vous pouvez utiliser le mot clé UNTIL (Jusqu'à). La différence est au niveau de la logique. Le WHILE exécute la boucle, tant que la condition est valide. Le UNTIL exécute la boucle jusqu'à ce que la condition soit valide :

```
1 i = 0
2 DO UNTIL i = 6
3     PRINT "J'ai ecrit " ; i ; " fois
          cette phrase"
4     i=i+1
5 LOOP
```

Ce code écrit la même chose que la boucle précédente, sauf que le programme, cette fois, attend jusqu'à ce que i soit égal à 6.

Il existe une dernière particularité. La condition peut, soit être :

- après le DO, comme nous l'avons vu jusqu'ici ;
- après le LOOP.

Si vous mettez la condition après le LOOP, vous êtes sûr que le programme exécute au moins une fois le code de la boucle.

Vous ne pouvez pas mettre de condition en même temps après le DO et le LOOP.

3.2.c Les procédures

3.2.c.a SUB

Le mot clé SUB permet de définir un bloc de code que vous pouvez exécuter en l'appelant avec CALL. Lorsque l'ordinateur arrive sur le mot clé CALL, il exécutera le contenu de ce qui est défini entre SUB et END SUB, puis continuera ce qui suit le CALL :

```
1 CALL direBonjour
2 PRINT "Au revoir tout le monde !"
3
4 SUB direBonjour
5 PRINT "Hello World"
6 END SUB
```

Les SUB doivent être placés en fin de code.

En réalité, le mot clé CALL n'est pas obligatoire, mais il rend le code plus clair.

3.2.c.b FUNCTION

Le mot clé FUNCTION est similaire au mot clé SUB. Il permet de définir un morceau de code à exécuter, mais lorsque vous utilisez FUNCTION, il est possible de retourner une valeur.

```
1  PRINT "Hello World"
2  retour = direAurevoir
3  PRINT "C'est fini"
4
5  FUNCTION direAurevoir
6  PRINT "Au revoir"
7  END FUNCTION
```

Ce qui affiche :

```
1  Hello World
2  Au revoir
3  C'est fini
```

Comme vous pouvez le remarquer, pour une fonction, il est nécessaire de récupérer une valeur (ici, dans la variable retour).

Tout comme pour les SUB, les FUNCTION doivent être définies en fin de code.

Ici, la variable retour aura pour valeur 0, car nous n'avons précisé aucune valeur de sortie.

Pour indiquer une valeur à retourner, il faut simplement définir une valeur à une variable du nom de la fonction :

```
1  retour = donneCinq
2  PRINT retour
3
4  FUNCTION donneCinq
5  donneCinq = 5
6  END FUNCTION
```

Ce qui affichera '5'.

3.2.c.c Les arguments

Pour rendre les SUB et les FUNCTION un peu plus polyvalents et réutilisables, il est possible de leur passer des arguments :

```
1  PRINT somme(10, 20)
2
3  FUNCTION somme (a, b)
4  somme = a + b
5  END FUNCTION
```

Ce code affichera '30'.

4 Exercice : jeu des allumettes

Pour apprendre la programmation et devenir bon dans ce domaine, il est important de pratiquer et de s'entraîner. Pour appliquer les quelques concepts vus précédemment, nous allons faire un petit jeu.

4.1 Règles

C'est un jeu à **deux joueurs**. **Quinze** allumettes sont posées sur la table et chacun des deux joueurs peut prendre **1, 2 ou 3** allumettes **tour à tour**. Celui qui prend la **dernière** allumette a perdu.

4.2 Réflexion

Lorsque vous avez un problème, quel qu'il soit, vous devez le décomposer et en garder l'essentiel. Ici, les mots clés ont été mis en gras. Il y a certes peu d'informations, mais l'exercice est simple.

Nous avons quinze allumettes au début. Donc :

```
1  nbAllumettes = 15
```

Le jeu ne se termine que lorsqu'un joueur a pris la dernière allumette. Cela sous-entend que le jeu fonctionne tant qu'il reste des allumettes donc nous pouvons inclure la partie, dans une boucle ayant pour condition :

```
1  DO WHILE nbAllumettes <> 0
2
3  LOOP
```

Ensuite, il faut que l'on sache combien d'allumettes le joueur veut. On utilisera donc INPUT, mais son choix est limité, donc, nous devons rajouter un test. Mais ce test ne sera pas un simple IF, sachant que s'il prend un choix invalide, nous devons lui faire choisir autre chose. Donc ce sera une boucle, tant qu'il n'a pas choisi le bon nombre :

```
1  allumettesPrises = 0
2      DO
3          INPUT "Combien d'allumettes
               prenez-vous " ;
               allumettesPrises
4      LOOP UNTIL allumettesPrises >= 1 AND
           allumettesPrises <= 3
```

Le test utilisé est allumettesPrises >= 1 ET allumettesPrises <= 3, il était possible aussi de faire : allumettesPrises = 1 OU allumettesPrises = 2 OU allumettesPrises = 3.

Toutefois, le test n'est pas complet. En effet, il est possible au joueur de prendre plus d'allumettes qu'il n'en reste sur la table. Nous ne voulons pas que ce soit le cas, donc nous rajoutons une condition :

```
1      DO
2          PRINT "Joueur " ; joueur
3          INPUT "Combien d'allumettes
               prenez-vous " ;
               allumettesPrises
4      LOOP UNTIL allumettesPrises >= 1 AND
           allumettesPrises <= 3 AND
           allumettesPrises <= nbAllumettes
```

Pour simuler le geste de « prendre des allumettes », il suffit de soustraire le nombre d'allumettes prises du nombre d'allumettes restantes :

```
1     nbAllumettes=nbAllumettes-
          allumettesPrises
2     PRINT "Il reste " ; nbAllumettes ; "
          allumettes"
```

Et on peut boucler.

Vous pouvez déjà tester le jeu, il est fonctionnel, mais pas parfait. En effet, il manque la gestion des joueurs. Ici, on ne fera rien de compliqué, simplement un joueur 1 et un joueur 2 qui s'alternent.

Donc, avant la boucle principale, on ajoutera :

```
1 joueur=1
```

On commence certes, par le joueur 1. C'est un choix arbitraire.

Pour un peu plus de clarté, nous pouvons rajouter :

```
1         PRINT "Joueur " ; joueur
```

dans la boucle lorsque le joueur sélectionne le nombre d'allumettes.

Ensuite, une fois que les allumettes sont retirées de la table, nous devons passer au tour de l'autre joueur.

Une simple incrémentation ne suffit pas, sinon, au bout du troisième tour, un troisième joueur apparaîtrait. Il existe néanmoins une astuce : nous avons deux joueurs, il suffit de soustraire la variable joueur à trois :

joueur

```
1 joueur=3-joueur
```

Vous pouvez tester :

- si joueur indique le premier joueur (et vaut 1), 3 - 1 = 2, nous sommes passés au second joueur ;
- si joueur indique le second joueur (et vaut 2), 3 - 2 = 1, nous repassons au premier joueur. Parfait !

Finalement, après la boucle principale, nous rajoutons :

```
1 PRINT "Joueur " ; joueur ; " a gagné"
```

4.3 Code final

```
1  PRINT "Jeu des allumettes en QB64 pour
       Developpez.com"
2
3  joueur=1
4  nbAllumettes = 15
5  PRINT "Vous avez 15 allumettes, vous
       pouvez en prendre soit 1, 2, ou 3"
6  DO WHILE nbAllumettes <> 0
7
8      allumettesPrises = 0
9      DO
10         PRINT "Joueur " ; joueur
11         INPUT "Combien d'allumettes
               prenez-vous " ;
               allumettesPrises
12     LOOP UNTIL allumettesPrises >= 1 AND
           allumettesPrises <= 3 AND
           allumettesPrises <= nbAllumettes
13
14     nbAllumettes=nbAllumettes-
           allumettesPrises
15     PRINT "Il reste " ; nbAllumettes ; "
           allumettes"
16
17     joueur=3-joueur
18 LOOP
19
20 PRINT "Joueur " ; joueur ; " a gagné"
```

4.4 Améliorations

Il est possible d'améliorer le jeu. Pour vous entraîner et mieux maîtriser QB64, vous pouvez :

- faire en sorte que le jeu demande si les joueurs veulent faire une nouvelle partie. Si oui, alors, on relance une partie (nbAllumettes = 15), sinon, on termine le programme ;
- placer la demande du nombre d'allumettes dans une fonction. La fonction doit accepter deux arguments, le joueur actuel et le nombre d'allumettes restantes et retourner le choix du joueur ;
- vous pouvez l'améliorer comme vous le souhaitez. Il est possible de changer la couleur d'affichage avec le mot clé COLOR.

*Retrouvez l'article d'**Alexandre Laurent** en ligne : lien 41*

Reseau

Les derniers tutoriels et articles

La technologie Ethernet

Ethernet est une technologie universelle qui dominait déjà les réseaux locaux bien avant le développement de l'Internet. La clé de la longévité de cette technologie, c'est sa simplicité. Souvent critiquée, elle a toujours été plus facile à utiliser et à mettre en œuvre que ses concurrentes. Cet article est à la fois une introduction aux normes (IEEE 802.3 - 10 Mbps, Fast Ethernet - 100 Mbps, Gigabit Ethernet - 1 Gbps, 10 Gbps) et une aide à la conception et la réalisation de réseaux locaux.

1 Copyright et Licence

1.1 Méta-information

Cet article est écrit avec DocBook XML sur un système Debian GNU/Linux. Il est disponible en version imprimable au format PDF : ethernet.pdf.

2 Ethernet : les raisons du succès

2.1 Quelques principes simples

- Toutes les stations sont égales vis-à-vis du réseau : il n'y a pas d'équipement maître de contrôle du réseau.
- La méthode d'accès employée est distribuée entre tous les équipements connectés.
- Le mode de transmission est de type bidirectionnel alterné : les signaux transitent dans les deux sens, mais pas simultanément.
- On peut relier ou retirer une machine du réseau sans perturber le fonctionnement de l'ensemble.

Ces principes ont montré qu'il était plus facile de concevoir les réseaux et les équipements correspondants avec Ethernet qu'avec d'autres technologies aux définitions plus complètes. De nombreuses technologies réseaux « mieux définies » au départ comme Token Ring (IEEE 802.5) par exemple, se sont avérées très peu évolutives au fil du temps.

Ces principes ont été formalisés au début des années soixante-dix. Aujourd'hui, seul le mode de transmission bidirectionnel alterné est de moins en moins employé. Le déploiement de la commutation de niveau 2 étant généralisé, les transmissions se font sur des paires cuivre ou fibre optique dédiées à chaque sens de communication. On parle alors de mode **full duplex**.

2.2 Ethernet a été intégré dans le modèle OSI

Ethernet était à l'origine un standard développé par les laboratoires Xerox au tout début des années 1970. Ce standard a d'abord évolué jusqu'à la version Ethernet II, aussi appelée DIX ou encore v2.0 avec l'association regroupant Digital Equipment Corporation, Intel et Xerox. Par la suite, Ethernet a été inclus dans les travaux sur la modélisation OSI au début des années 1980. Depuis cette époque, la technologie Ethernet est totalement in-

dépendante des constructeurs ; c'est un des facteurs importants de sa popularité.

Les éléments de la couche physique (couche 1 OSI) sont définis par les normes IEEE des sous-comités 802.3 et la méthode d'accès CSMA/CD correspond à la partie MAC de la couche liaison (couche 2 OSI).

Comme dans le cas des principes énoncés ci-avant, la généralisation de la commutation simplifie la méthode d'accès en éliminant toute la partie consacrée à la gestion des collisions. On attache aujourd'hui beaucoup plus d'importance aux méthodes de codage employées au niveau de la couche physique.

2.3 Une évolution constante

La simplicité de la méthode d'accès et la simplicité de l'interconnexion avec les autres technologies ont fait d'Ethernet une technologie évolutive à des coûts acceptables pour toutes les catégories d'utilisateurs.

Même si les évolutions des débits ont entraîné l'abandon de supports bon marché (câbles coaxiaux lors du passage de 10 à 100 Mbps), la mise en œuvre est restée simple. Les infrastructures existantes progressent vers les technologies multimédias sans réinvestissements lourds.

C'est une des grandes leçons de l'histoire des réseaux de télécommunications sur les trente dernières années. Toutes les technologies de transmission qui ont cherché à qualifier les flux réseau au plus près du matériel n'ont pas su évoluer simplement. L'exemple de la technologie ATM est caractéristique. Faire évoluer les équipements actifs ATM pour adapter les débits est excessivement plus coûteux qu'avec des équipements Ethernet.

Au début des années 1970 :

— le premier réseau local Ethernet expérimental a été développé au centre de recherche Xerox de Palo Alto (PARC) pour interconnecter des ordinateurs et des imprimantes laser à un débit de 2.94 Mbps. En juillet 1976, les deux concepteurs de ce réseau, Bob Metcalfe et David Boggs, publièrent le document de référence Ethernet : Distributed Packet Switching for Local Computer Network.

En septembre 1980 :

— le premier standard Ethernet est publié. Les sociétés Intel et Digital Equipment Corporation (DEC) ont rejoint Xerox pour produire un standard utilisable par tous. On a baptisé ce standard DIX standard. Il correspond à la version 10Base5 ou Ethernet « épais ». Voir Section 4.1.1, « Ethernet standard ». Les premières cartes Ethernet sont apparues avec la version 2.0 du standard DIX en Novembre 1982 : le standard Ethernet II.

En 1983 :

— la première norme Ethernet est publiée par l' Institute of Electrical and Electronic Engineers (IEEE) ; plus précisément par le sous-comité IEEE 802.3. C'est à cette époque qu'est apparue la double signification d'un champ dans le format de la trame Ethernet : le champ Type/Longueur. Cette différence entre normalisation et standard n'a jamais eu d'effet sur l'exploitation des réseaux locaux Ethernet. Voir Section 9, « Format de trame » ;

— en 1985, l'IEEE publia la norme IEEE 802.3a correspondant à l'Ethernet « fin ». En 1987, l'utilisation des fibres optiques devint effective avec la norme IEEE 802.3d.

En 1990 :

— la première norme utilisant les câbles de paires torsadées cuivre sur une topologie étoile est publiée : IEEE 802.3i. C'est à partir de cette étape que les autres technologies de réseaux locaux ont décliné rapidement.

En 1993 :

— la norme IEEE 802.3j est venue étendre l'application de la topologie étoile sur fibres optiques.

En 1995 :

— nouvelle étape majeure dans l'évolution d'Ethernet : le passage à 100 Mbps avec l'introduction de la norme IEEE 802.3u. Cette version d'Ethernet est connue sous le nom Fast Ethernet.

En 1997 :

— la norme IEEE 802.3x a défini le mode « full-duplex » qui permet de réserver une paire cuivre ou fibre optique par sens de communication. Associée à la généralisation de l'utilisation des commutateurs, cette norme marque la fin de l'utilisation de la méthode d'accès historique d'Ethernet : CSMA/CD.

En 1998 :

— les débits ont à nouveau été multipliés par 10 avec la sortie du Gigabit Ethernet. La norme correspondante est l'IEEE 802.3z ;

— cette première définition a été complétée en 1999 avec la norme IEEE 802.3ab qui définit l'utilisation du Gbps sur les câbles en paires torsadées UTP de catégorie 5.

En 2002 :

— une fois de plus, les débits ont été multipliés par 10 pour atteindre les 10 Gbps avec la publication de la norme IEEE 802.3ae. Cette catégorie de débit marque l'avènement de l'exploitation d'Ethernet sur les dorsales des réseaux étendus ;

— de même qu'avec le Gigabit Ethernet, une définition d'Ethernet 10 Gbps sur paires torsadées cuivre devrait voir le jour prochainement. La norme devrait être publiée avec l'appellation IEEE 802.3an.

L' Institute of Electrical and Electronic Engineers Get IEEE 802 a mis à disposition en ligne les normes du comité 802 : . Ce document est construit à partir des quatre familles de débits d'Ethernet : Ethernet à 10 Mbps : la définition d'origine à partir de la constitution du sous-comité IEEE 802.3.

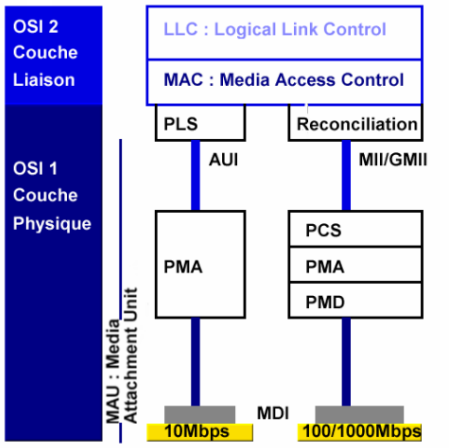
— Ethernet à 100 Mbps ou Fast Ethernet.

— Ethernet à 1 Gbps ou Gigabit Ethernet.

— Ethernet à 10 Gbps ou 10 Gigabit Ethernet.

À l'intérieur de chaque famille, il existe de nombreuses déclinaisons. Les plus utilisées sont décrites ci-après. Du point de vue de la conception, les câblages en paires torsadées cuivre sont habituellement utilisés pour la « desserte » des postes de travail à des débits allant de 10 Mbps à 1 Gbps. Ensuite, les câblages en fibres optiques sont utilisés pour les dorsales réseau. Bien que cela ne corresponde à aucune normalisation, on rencontre de plus en plus souvent un découpage en trois couches lors de la conception des réseaux locaux Ethernet : accès, distribution et cœur. Ce découpage a surtout pour but de faciliter le classement des équipements dans les catalogues constructeurs. Pour en savoir plus sur la hiérarchie dans les réseaux locaux, lire l'article : Segmentation des réseaux locaux.

3 Définitions IEEE 802.3



Cette vue met en évidence les éléments introduits pour faire évoluer les débits.
Les sigles :

— AUI : Attachment Unit Interface ;

— MDI : Media Dependant Interface ;

— MII : Media Independant Interface : reconnaissance des vitesses 10/100/1000 Mbps ;

— PCS : Physical Coding Sublayer ;

— PLS : Physical Layer Signaling ;

— PMA : Physical Media Attachment sublayer ;

— PMD : Physical Media Dependant sublayer ;

— IV Normalisations IEEE 802.3

4 Normalisations IEEE 802.3

Il existe de nombreux suppléments à la norme IEEE 802.3 initialement publiée, qui décrivent les différents supports utilisables.

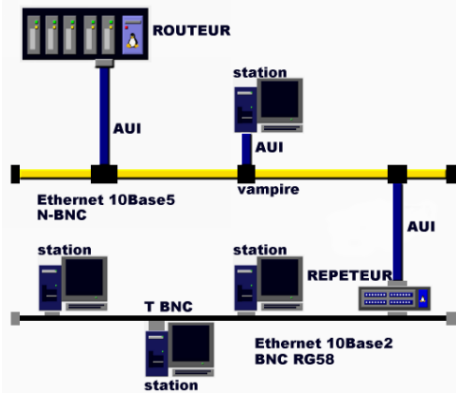
4.1 Ethernet IEEE 802.3

C'est le point de départ de la normalisation. La première définition est la plus proche du *Standard Ethernet II* publié par DEC, Intel et Xerox.

La topologie utilisant des câbles coaxiaux est toujours de type BUS. Cette topologie était avantageuse lorsque le nombre et la disposition des stations changeaient. Aujourd'hui, les câbles coaxiaux sont systématiquement abandonnés au profit des câbles en paires torsadées cuivre ou des fibres optiques. Le coût de la connectique des câbles coaxiaux est devenu supérieur à celui de la connectique RJ45 utilisée avec les paires torsadées.

4.1.a Ethernet standard

Le câble standard a été défini à l'origine pour des connexions avec **transceivers** à piquage (vampire) puis étendu à la connectique de type N-BNC.



Appellation	10Base5, Thick Ethernet
Support	câble coaxial 50 ohms associé à une connectique N-BNC
Longueur maximum	500 m par brin. Les câbles doivent avoir une longueur multiple de 23,4 m (généralement 117 m) pour que les réflexions produites sur les raccords soient superposées déphasées
Distance entre connexions	au moins 2,50 m (points repérés sur le câble)
Nombre maximum de connexions	au plus 100 connexions par brin

Tableau 1. Ethernet Standard

4.1.b IEEE 802.3 a

Le câble standard a été défini à l'origine pour des connexions avec **transceivers** à piquage (vampire) puis étendu à la connectique de type N-BNC.

Appellation	10Base2, Thinnet ou Thin Ethernet
Support	câble coaxial 50 ohms (RG58) associé à une connectique BNC
Longueur maximum	185 m
Distance entre connexions	50 cm
Nombre maximum de connexions	30 stations

Tableau 2. Ethernet fin

4.1.c IEEE 802.3 c-d

Définit les caractéristiques des répéteurs 10Base2 ainsi que les liaisons FOIRL (Fiber Optic Inter Repeater Link) fibre optique entre les répéteurs. Le répéteur interconnecte les brins de média entre eux en :

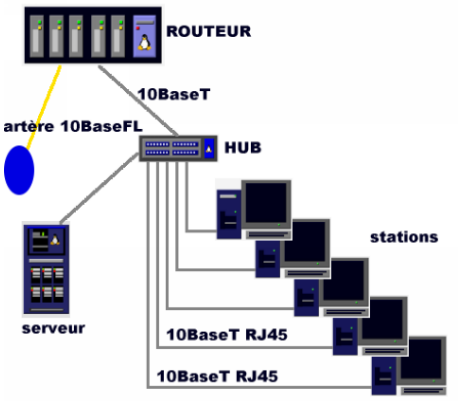
- régénérant les signaux ;
- prolongeant les fragments (morceaux de trames issus des collisions) ;
- complétant les préambules.

Il peut aussi intervenir sur la propagation des collisions (Jamming) ou interrompre une émission trop longue.

La liaison FOIRL doit avoir une longueur inférieure ou égale à un km.

4.1.d IEEE 802.3 i

Introduite en 1990, cette définition constitue une évolution majeure d'Ethernet. C'est la première à adopter une topologie étoile analogue à celle des installations téléphoniques. Depuis, cette topologie étoile domine très largement dans les installations réseau.



Appellation	10BaseT débit 10 Mbps
Support	paire torsadée non blindée (UTP : Unshielded Twisted Pair) associée à une connectique RJ45 en topologie étoile.
Longueur maximum	100 m

Tableau 3. 10BaseT

4.1.e IEEE 802.3 j

Cette définition a très largement été utilisée pour l'implantation des dorsales réseau de campus.

Appellation	10Base-F débit 10 Mbps
Support	fibre optique multimodes (62.5/125 μm) associée à une connectique ST ou SC.
Longueur maximum	2 km

Tableau 4. 10BaseT

10BASE-FL : redéfinition de FOIRL avec des capacités plus intéressantes telles que la possibilité de concevoir une topologie étoile avec des répéteurs multiports.

5 Fast Ethernet IEEE 802.3u

Publiées en 1995, ces spécifications ont très vite été adoptées. Le coût par port a chuté de 50 % entre 1996 et 1999.

5.1 100BaseT

La signalisation 100Base-X sur les câbles et fibres reprend celle développée pour la technologie FDDI (Fiber Distributed Data Interface).

Appellation	100BaseT débit 100 Mbps
Support 100Base-T4	utilise 4 paires (transmission, réception, 2 bidirectionnelles) de câbles UTP de catégories 3, 4 ou 5. Les 100 Mbps sont répartis sur 3 paires.
Support 100Base-TX	utilise 2 paires (transmission, réception) de câbles UTP5 ou STP (Shielded Twisted Pair). Ce câble supporte 200 Mbps en mode full duplex après négociation entre les extrémités.
Longueur maximum	100 m

Tableau 5. 100BaseT

5.2 100BaseF

Appellation	100BaseFX débit 100 Mbps
Support	fibre optique multimode (62.5/125 μm) associée à une connectique ST ou SC.
Longueur maximum	400 m

Tableau 6. 100BaseFX

6 Gigabit Ethernet

Comme les câbles en paires torsadées de catégorie 5 sont certifiés pour des fréquences allant jusqu'à 100 MHz (cf TIA/EIA-568-A), le passage à 1000 Mbps pose des difficultés nouvelles par rapport aux évolutions précédentes. La couche physique a été entièrement revue. La nouvelle définition est une « fusion » de deux technologies : l'Ethernet IEEE802.3 et le Fiber Channel ANSI X3/T11.

Cette fusion reprend le format de trame Ethernet 802.3 et la méthode d'accès CSMA/CD full-duplex, pour conserver la compatibilité avec les couches supérieures du réseau, et elle bénéficie du débit élevé de l'interface physique Fiber Channel. Comme pour la famille Fast Ethernet, il existe plusieurs variantes 1000BaseX.

6.1 définitions IEEE 802.3z : 1000BaseX

Appellation	1000BaseLX
Support	laser grandes ondes sur fibre optique multimode et monomode destiné aux artères de campus.
Longueur maximum	3 km

Tableau 7. 1000Base-LX

Appellation	1000BaseSX
Support	laser ondes courtes sur fibre optique multimodes destiné aux artères intra-muros.
Longueur maximum	500 m

Tableau 8. 1000Base-SX

Appellation	1000BaseCX
Support	câble en paires torsadées blindées 150 ohms destiné aux connexions entre serveurs dans le même local.
Longueur maximum	25 m

Tableau 9. 1000Base-CX

6.2 définition 1000BaseT : IEEE802.3ab

Cette définition est très importante. C'est elle qui permet d'utiliser le Gigabit Ethernet dans la majorité des installations actuelles. Ceci dit, les installations existantes auront certainement besoin d'une « requalification » avant d'être équipées en 1000BaseT. Cette technologie utilise les câbles FTP de ca-

tégorie 5 au maximum de leur certification. De nouvelles catégories de câbles sont en cours de spécification : 5enhanced à 100 MHz, 6 à 200 MHz et 7 à 600 MHz. Il est recommandé de limiter au maximum les brassages intermédiaires dans les armoires de câblage.

Appellation	1000BaseT
Support	câble en paires torsadées non blindées de catégorie 5.
Longueur maximum	100 m

Tableau 10. 1000Base-T

7 Gigabit Ethernet

7.1 IEEE 802.3ae

Définition 10 Gbps

8 Négociation de la bande passante

Avec les évolutions des débits d'Ethernet, la grande majorité des infrastructures actuelles sont « mixtes ». Il est donc nécessaire que les équipements réseau (cartes, commutateurs, etc.) soient capables de reconnaître et d'utiliser la bande passante disponible sur le câble(1).

8.1 Autonégociation

La négociation entre équipements porte sur deux fonctionnalités :

— le débit 10, 100 et 1000 Mbps,

— le mode half-duplex ou full-duplex (IEEE 802.3x).

La fonction d'autonégociation est optionnelle. Elle est apparue avec l'extension Fast Ethernet et ne concerne que les câbles en paires torsadées et les fibres optiques. La fonction de négociation utilise les signaux de contrôle d'état du lien physique en respectant la compatibilité entre tous les équipements indépendamment de leur génération et des options qu'ils supportent.

L'ordre des négociations est le suivant :

— 100BaseTX Full-Duplex ;

— 100BaseT4 ;

— 100BaseTX ;

— 10BaseT Full-Duplex ;

— 10BaseT.

8.2 Mode Full-Duplex IEEE 802.3x

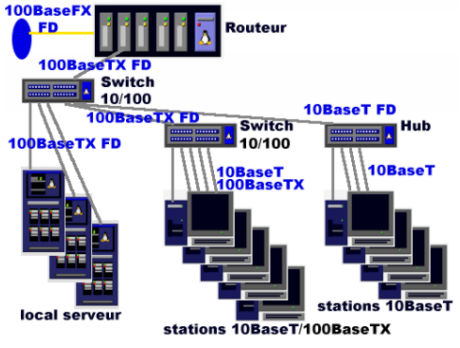
Le mode de fonctionnement de base d'Ethernet est dit *Half-Duplex* : bidirectionnel à l'alternat. Il suppose que le média est partagé entre plusieurs stations et que les informations transitent dans les deux sens. C'est sur ce mode de base que la méthode d'accès CSMA/CD est bâtie.

Le mode Full-Duplex correspond à une communication point-à-point entre deux équipements. Dans ce contexte, le média n'est plus partagé entre plus de deux stations et les informations transitent toujours dans les deux sens mais sur des canaux (paires torsadées ou fibres) distincts.

En conséquence, l'algorithme d'accès au média est considérablement simplifié et la bande passante utile est doublée. Aujourd'hui, beaucoup de constructeurs ont adopté ces options d'Ethernet. Il est donc possible de concevoir des artères à 200 Mbps entres commutateurs et/ou concentrateurs (hubs) pour un coût très raisonnable.

8.3 Exemple de conception

L'exemple qui suit est une synthèse sur l'utilisation de la technologie Ethernet. Il ne prend pas en compte certaines autres normes telles que les VLAN IEEE 802.1Q qui autoriseraient d'autres modes de conception.



— 100BaseFX-FD : dorsale de campus en fibres

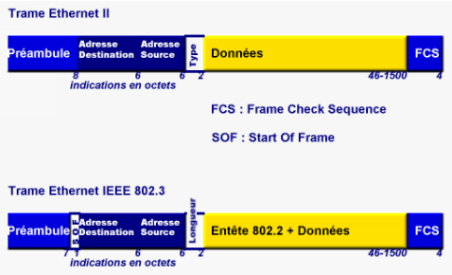
optiques avec un débit de 200 Mbps en Full-Duplex.

— 100BaseTX-FD : alimentation du commutateur mixte (Switch 10/100) du local serveur (Server Farm) avec un débit de 200 Mbps en Full-Duplex.

— 100BaseTX-FD : alimentation du commutateur mixte (Switch 10/100) des stations 10/100 (typiquement multimédia/CAO) avec un débit de 200 Mbps en Full-Duplex.

— 10BaseT-FD : alimentation du commutateur (Switch) des stations 10BaseT (typiquement bureautiques) avec un débit de 20 Mbps.

9 Format de trame



La signification de chacun des champs de trame est donnée ci-après.

10 Le préambule

Le préambule est une suite de 0 et de 1 alternés. Il permet à l'horloge du récepteur de se synchroniser sur celle de l'émetteur. Comme la transmission est asynchrone, il est possible qu'une partie du préambule soit perdue.

Même si la norme IEEE 802.3 a spécifié un champ spécifique en fin de préambule : SOF (Start of Frame) avec deux bits à 1, il n'y a aucune différence avec le standard Ethernet v2.0 pour lequel les deux derniers bits du préambule sont aussi à 1.

10.1 Les adresses MAC

Les adresses MAC identifient le ou les destinataire(s) de la trame puis l'émetteur. Elles sont constituées de six octets :

— les trois premiers octets font référence au constructeur de l'interface. Ils sont uniques et sont attribués par l'IEEE ;

— les trois octets suivants donnent le numéro d'interface chez ce constructeur.

L'adresse source est toujours celle d'une interface unique (unicast). La destination peut être une adresse unique, de groupe (multicast) ou de diffusion générale (broadcast = FF-FF-FF-FF-FF-FF). Dans une adresse de groupe, le premier bit transmis est à 1. Si les autres bits ne changent pas, l'adresse de groupe correspond à toutes les cartes d'un même constructeur.

10.2 Le champ longueur/type

Ce champ de deux octets a été défini dans le standard Ethernet II pour indiquer le type de protocole de niveau 3 employé pour transmettre le message.

Avec la normalisation originale IEEE 802.3, ce champ a été redéfini pour contenir la longueur en octets du champ des données.

Depuis 1997, la normalisation IEEE intègre les deux formats de trames. Parallèlement, les documents standards des protocoles de l'Internet se sont appuyés sur des trames Ethernet utilisant des champs types (Voir RFC895 de 1984). À l'heure actuelle on peut considérer que les trames IEEE avec champ type correspondent au trafic utilisateur sur les réseaux IP et que les trames IEEE avec champ longueur correspondent au trafic de dialogue entre équipements actifs (Algorithme STP, etc.).

10.3 Les données

Ethernet II

— D'après la définition d'origine, la couche 2 est complète avec ce format. Les données sont directement transmises au niveau réseau identifié par le champ type. Aucune « séquence de bourrage » ou padding n'est prévue bien que le nombre minimum de données attendues soit de 46 octets.

IEEE 802.3

— Le champ de données contient l'en-tête de la sous-couche LLC en plus des données. Au niveau MAC, ce champ est vu comme une suite de 46 à 1500 octets que l'on n'interprète pas.

— Si le nombre de données n'atteint pas 46 octets, le champ est complété par une séquence de bourrage (padding).

10.4 Le champ de contrôle

Le FCS : Frame Check Sequence est un champ de quatre octets qui permet de valider l'intégrité de la trame à un bit près. Il utilise un CRC (Cyclic Redundancy Check) qui englobe tous les champs de la trame. Ainsi, la station réceptrice peut décider si la trame est correcte et doit être transmise à la couche supérieure : LLC (Logical Link Control IEEE 802.2)

ou réseau. Voir Section 11.2, « Sous-couche LLC : IEEE 802.2 ».

10.5 Le temps inter-trame

Le temps inter-trame est appelé indifféremment *Inter Frame Space* ou *Inter Frame Gap*.

Une machine ne peut émettre toutes les trames qu'elle a à transmettre les unes à la suite des autres. Le délai inter-trame normalisé est de 96 bits soit 9,6 microsecondes à 10 Mbps. Attention, cette définition a été revue pour le Gigabit Ethernet. Voir Section 6.3, « Extension CSMA/CD ». Il correspond au temps minimum de retour au repos du média et permet aux autres stations de prendre la main.

11 Trames erronées

Préambule	Adresse destination	Adresse source	Type/Longueur	LLC	Données	FCS

Tableau 11. Les champs de trame

À la suite d'incidents tels qu'une collision, le débranchement brutal d'une machine, la perte du bouchon d'adaptation d'impédance ou le mauvais fonctionnement d'une partie du matériel, des trames non cohérentes peuvent apparaître. Certains de ces défauts sont répertoriés avec un vocabulaire particulier.

12 Runt

Ce terme désigne les trames trop courtes (moins de 64 octets). Ce type de trame est le plus souvent le résultat d'une collision.

12.1 Jabber

Il s'agit d'une trame trop longue (plus de 1518 octets). On distingue deux types de causes :

— s'il y a superposition de deux trames sans détection de collision, on peut considérer que les couches 1 et 2 d'une interface du réseau ne fonctionnent plus correctement ;

— la trame n'a plus de structure et est émise par un composant qui reste beaucoup trop longtemps en émission.

Ce type de défaut est très pénalisant pour le réseau et entraîne une dégradation rapide des performances.

12.2 Misaligned frame

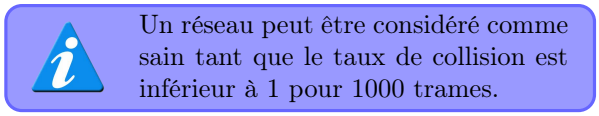
Une trame désalignée est une trame dont le nombre de bits n'est pas divisible par 8. Dans la pratique, ce type de trame possède presque toujours un CRC faux.

12.3 Bad FCS

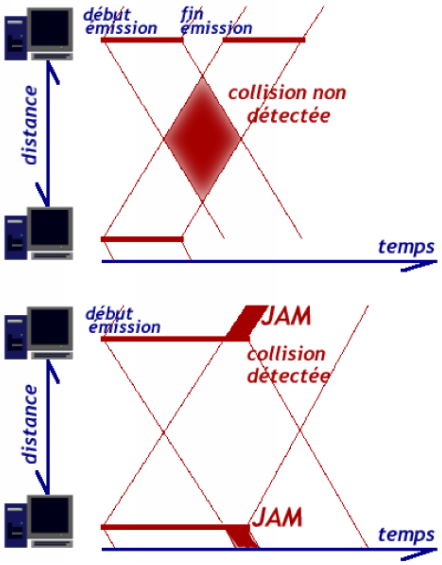
Il s'agit d'une trame complète dont un bit n'a pas été reçu tel qu'il avait été transmis ou d'une trame tronquée résultant d'une collision.

13 Les collisions

Ce phénomène résulte de la superposition de deux trames sur le média lorsque deux stations émettent simultanément.



Un réseau peut être considéré comme sain tant que le taux de collision est inférieur à 1 pour 1000 trames.



13.1 Les collisions tardives

Ce type de collision intervient lorsque la longueur du câble dépasse la norme ou lorsqu'il y a trop de répéteurs dans le réseau.

C'est le seul type de collision que l'on rencontre sur les réseaux câblés en paires torsadées et constitués de commutateurs et de routeurs. Les « collisions » apparaissent avec des segments de plus 100 m.

14 Couche liaison et Ethernet

14.1 Sous-couche MAC : méthode d'accès CSMA/CD

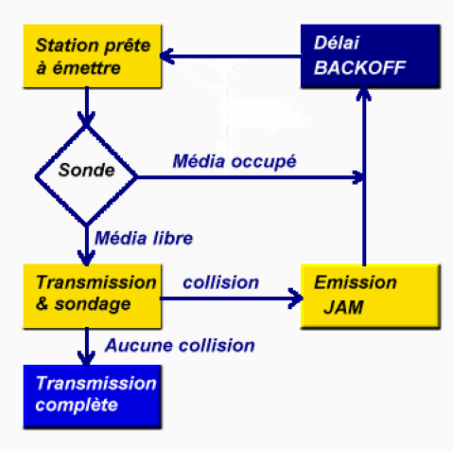
La méthode CSMA/CD est dérivée d'un système de transmission radio appelé Aloha. Son principe est de laisser chacun libre de gérer ses émissions en fonction de ses besoins et de la disponibilité du média.

En l'absence d'information à transmettre, la station écoute (ou reçoit) les paquets qui circulent sur le câble dans un sens ou dans l'autre. Quand la station a besoin d'émettre un ou plusieurs paquets, elle agit indépendamment des autres. Elle sait juste que lorsqu'elle perçoit une trame, une autre machine doit être en émission.

Chaque machine ayant à tout instant la possibilité de débuter une transmission de manière autonome, la méthode d'accès est distribuée : elle est dite à accès multiple (Multiple Acess : MA). La machine observe le média en cherchant à détecter une porteuse (Carrier Sense : CS). Si aucune trame n'est en transit, elle ne trouve pas de porteuse. Elle envoie ses paquets sur le support physique et reste à l'écoute du résultat de son émission pendant quelque temps, pour vérifier qu'aucune autre machine n'a suivi le même comportement qu'elle au même instant.

La méthode d'accès étant à détection de collision (Collision Detect : CD), lors de son émission une machine peut déceler un problème de contention et s'arrêter avec l'intention de renvoyer son paquet ultérieurement quand elle aura de nouveau la parole. De façon à minimiser le risque de rencontrer une deuxième collision avec la même machine, chacune attend pendant un délai aléatoire avant de tenter une nouvelle émission.

Cependant, de manière à ne pas saturer un réseau qui s'avérerait déjà très chargé, la machine n'essaiera pas indéfiniment de retransmettre un paquet. Si à chaque tentative elle se trouve en conflit avec une autre ; après un certain nombre d'essais infructueux, le paquet est éliminé. On évite ainsi l'effondrement du réseau. Les couches supérieures sont averties que la transmission du message a échoué.



14.2 Sous-couche LLC : IEEE 802.2

À partir de cette sous-couche, on sort du domaine d'appellation Ethernet. Cependant, de nombreux réseaux locaux associent la norme IEEE 802.2 avec Ethernet.

Le sous-comité IEEE 802.2 a standardisé une couche de niveau LLC qui possède plusieurs types d'opérations offrant des services de différentes qualités.

- Le premier type d'opération est un service minimum, sans connexion (pas de liaison logique) ni acquittement (pas de retour d'information sur le déroulement de l'acheminement). Le type 1 permet des communications en point à point (un émetteur, un récepteur) ou en diffusion (un émetteur, plusieurs récepteurs).
- Le type d'opération 2 est un service sur connexion (liaison logique entre SAP) avec acquittement, vérification de l'ordre des trames, détection et correction d'erreur, détection des doubles, contrôle de flux. L'identificateur correspondant au couple SSAP/DSAP (Source Service Access Point/Destination Service Access Point) est unique. Ce type d'opération ne permet que des communications en point à point.
- Le type d'opération 3 est un service datagramme (sans connexion) acquitté, sans retransmission (pas de correction des erreurs), réalisant une prestation de qualité intermédiaire à la fois simple et performante.

Dans tous les cas, quel que soit le type d'opération, les données du niveau LLC sont présentées sous la forme d'un LPDU (LLC Protocol Data Unit), tel que représenté ci-dessous.

Adresse DSAP	Adresse SSAP	Control	Données

Tableau 12. Les champs de trame

15 En guise de conclusion

Cette synthèse sur la technologie Ethernet est nécessairement incomplète et certains aspects ont été volontairement occultés. En voici deux exemples significatifs :

La signalisation. Dans le domaine de l'instrumentation réseau, on ne mélange pas le test de câble et l'analyse de protocole. Il en est de même pour cette présentation, l'aspect signalisation dans la couche physique n'a pas été traité. C'est un sujet à part entière qui mérite que l'on s'y intéresse. Après avoir connu une décennie plutôt « tranquille », le test de câble va probablement revenir au devant de l'actualité avec l'utilisation du Gigabit Ethernet sur les câbles de catégorie 5 qui sont spécifiés à 100 MHz seulement.

Les autres définitions 100 Mbps. Il en existe deux : 100BaseT2 et 100BaseVG AnyLan. Pour le 100BaseT2, la question est vite réglée, il n'existe aucun équipement qui l'utilise à ma connaissance. La situation est à peine différente pour le 100BaseVG AnyLan, cette définition a été publiée en 1995 en même temps que celle du 100BaseTX. Sans rentrer dans la polémique sur le fait qu'elle ne respecte pas la méthode d'accès CSMA/CD, la base d'équipements 100BaseVG AnyLan installée est aujourd'hui insignifiante comparée à la base 100BaseTX.

Retrouvez l'article de ***Philippe Latu*** *en ligne : lien 45*

Java

Les dernières news Java

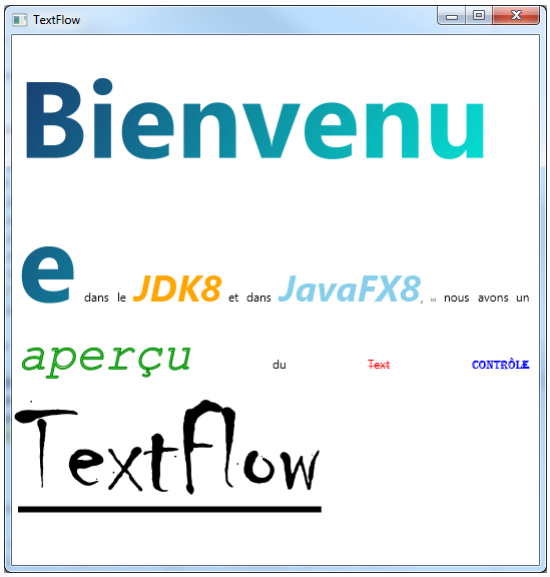
Java 8 apporte aussi des nouveautés du côté de Java FX, venez découvrir les améliorations de JavaFX 8

Java 8 est sortie le 18 mars dernier. Avec cette nouvelle mouture, nous avons pu voir d'importantes nouveautés au sein de la discussion Java 8 est disponible, la plate-forme se met aux expressions lambdas, tour d'horizon des nouveautés. Celles-ci concernaient aussi bien le langage, son API ou encore la machine virtuelle.

Intéressons-nous maintenant à l'aspect interface graphique. Si AWT, Swing ou encore Java2D n'ont pas subi de grosses évolutions en dehors de correctifs de bug, JavaFX dispose de nombreuses nouveautés. La première concerne un alignement des versions : adieu JavaFX 2.x, place à JavaFX 8.

Les nouveautés de l'api

— **Texte riche** : possibilité d'ajouter des styles et des effets aux textes via la classe TextFlow



Chaque mot, espace compris, est affiché via une instance de Text, chacune avec son style CSS propre. Le contrôle parent TextFlow fait automatiquement la mise en page correcte. Ici, le contrôle est configuré pour afficher un paragraphe justifié.

— **Support de Swing** : cette fonctionnalité disponible dans JavaFX 1.x et qui permet d'inclure des composants Swing dans une interface à base de contrôles JavaFX, n'avait pas été retenue dans la version 2

— **Skin** : son API est désormais totalement publique

— **Gestion de l'impression** : plus besoin de passer par AWT ou Swing pour gérer l'impression, ceci causait des problèmes sur MacOS, car JavaFX provoquait le plantage de Swing.

— **Les améliorations WebView** : support de WebWockets et WebWorker

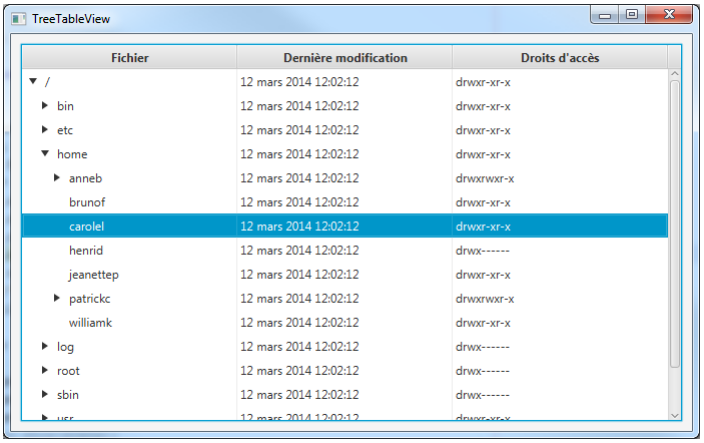
— **Retraits des builders de l'API** : ils sont dépréciés et seront supprimés dans Java 9

— **Communication AWT/Swing et JavaFX** : il devient possible via la ligne de commande de faire que l'EDT (Thread d'affichage et de propagation des événements) de AWT/Swing et le JavaFX Application Thread de JavaFX se fondent en un seul thread. Cette fonctionnalité rend ainsi les dialogues entre les différents composants plus aisés à développer (plus besoin alors de SwingUtilities.invokeLater() et Platform.runLater()).

Les nouveautés concernant le CSS

— **Nouvelle API** : celle-ci permet de définir des propriétés directement dans votre code Java

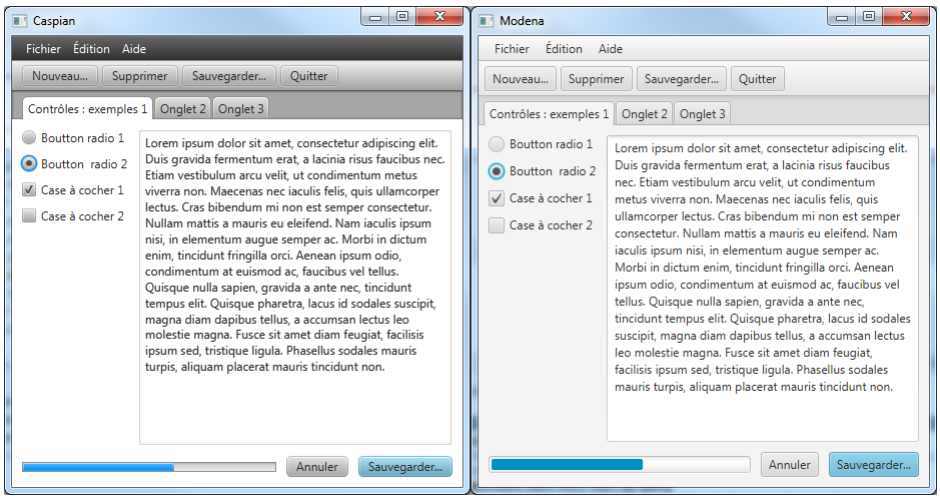
— Thème Modena : le CSS de base Caspian est remplacé par Modena



Composant TreeTableView permettant d'afficher une table dont la première colonne est un arbre.

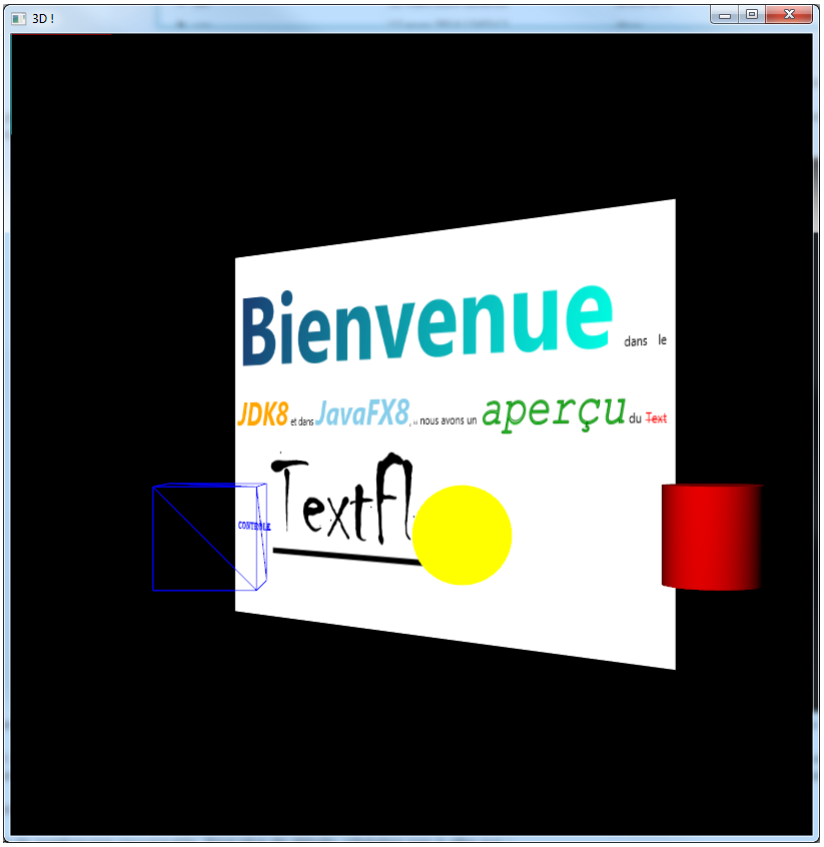
Les nouveautés graphiques

— **Des nouveaux composants graphiques** : ajout d'un DatePicker (calendrier) et d'un TreeTableView (tableau dont la première colonne contient un arbre). Nous regretterons le manque d'un Spinner et d'un FormattedTextField.



Caspian à gauche remplacé par Modena à droite.

— **Gestion de la 3D** : avec l'arrivée des primitives 3D, le support des meshes (pour importer des fichiers créés dans des applications externes), les textures, la gestion de l'éclairage la possibilité de rendre des sous-parties d'une scène ou encore de déplacer la caméra, l'on peut donc dire que la 3D est vraiment utilisable au sein de JavaFX.



Box avec rendu fil de fer, Sphere avec rendu ambiant, Cylinder avec rendu spéculaire. Les contrôles 2D peuvent également être positionnés et transformés dans l'espace 3D. Le support de l'anticrénelage a été activé sur la scène.

En complément, le JAR de runtime de JavaFX est désormais dans le répertoire lib/ext du JRE (mécanisme des extensions) et donc toujours sur le ClassPath. Nous pouvons également préciser que JavaFX fait partie du profil embedded (sans WebView ou le support Media). Cette fonctionnalité sera donc disponible dans les JVM pour les Raspberry Pi et autres. Prévu pour cette release, le support de l'enregistrement des Media devrait venir dans une mise à jour ultérieure.

Rappelons également que le passage à l'OpenSource est terminé depuis la fin de l'année 2013.

Comme vous pouvez le voir, JavaFX présente de nombreuses nouveautés. Pour plus de détails, n'hésitez pas à aller sur :

— La documentation officielle : lien 46

— Téléchargez JavaFX 8 : lien 47

Commentez la news de ***Mickael Baron*** *en ligne : lien 48*

Scala passe à la version 2.11

Le langage de programmation introduit un support expérimental pour Java 8 et des correctifs pour 600 bogues

Après plusieurs mois de développement depuis la version majeure 2.10, l'entreprise Typesafe dédiée au développement du langage Scala vient d'annoncer la sortie de la version 2.11.0.

Cette dernière version cible Java 6, mais introduit cependant un support expérimental pour Java 8 qui se limite actuellement à la lecture du bytecode et à l'analyse grammaticale du code source. Toutefois, les prochaines mises à jour tenteront d'améliorer ce support.

En outre, la version 2.11 apporte des correctifs à plus de 600 bogues, dont certains sont rétrocompatibles avec la version 2.10, même si cette version est passée en mode maintenance et sera arrêtée à la fin de l'année 2014, avec la sortie de Scala 2.10.5.

Cette version ne s'arrête pas uniquement à ces quelques correctifs. Ses auteurs la décrivent comme une version à la fois plus légère, plus rapide/performante et enfin plus stable. Sa légèreté se traduit par le déplacement d'un cinquième du code source de la bibliothèque principale vers des modules externes. Quant à la rapidité soulignée par les auteurs, elle est due à l'optimisation des performances du compilateur avec une génération plus rapide du code en utilisant le backend expérimental GenBCode et à l'implémentation de mécanismes permettant l'élimination des branches via une analyse de tout temps. Cette version se veut aussi plus stable à travers le refactoring de plusieurs collections, la correction de bogues existants sur l'IDE, mais aussi la modification de l'API expérimentale Reflection API.

Pour ceux d'entre vous qui souhaitent utiliser leurs macros (créées sous 2.10.x) avec la 2.11.0 un document a été publié pour expliquer la démarche à suivre.

Enfin, Jason Zaugg membre de l'équipe Scala de Typesafe a passé en revue toutes les modifications apportées à cette version dans ce webinar : lien 49.

*Commentez la news d'**Arslan Hamza Cherif** en ligne : lien 50*

Les derniers tutoriels et articles

Tutoriel Ceylon : concepts de base

Nombreux sont les langages dédiés à la JVM, mais aucun autre que Ceylon ne propose un système de typage aussi poussé ainsi qu'une compilation en JavaScript. À travers une série d'articles, je propose de vous faire découvrir tous les mystères de ce langage conçu par Gavin King (lien 51).

1 Avant-propos

Cet article est le deuxième d'une série articulée sur la présentation du langage Ceylon :

- Présentation et installation : lien 52 ;
- Concepts de base ;
- Typage ;
- Appels et arguments ;
- Collections ;
- Modules ;
- Interopérabilité avec Java.

Cet article est dédié à la présentation des éléments de base du langage : les littéraux, les types de base, les structures de contrôle et la déclaration de types.

2 Modules

2.1 Présentation

Dans Ceylon, l'élément de plus haut niveau est le « module » qui délimite une librairie ; c'est l'équivalent des .so/.dll (natifs ou de .Net) ou des .jar (de Java). Ces derniers sont constitués d'un ensemble de « packages », l'équivalent des « namespaces » du C++/C#. Et enfin viennent les « déclarations ». Les fichiers servent uniquement à répartir les déclarations selon vos goûts et vos couleurs. Ainsi, si vous créez les deux fichiers de source ci-dessous, vous obtiendrez une erreur de compilation vous signalant que maFonction est déclarée deux fois :

Doublon (Première déclaration)

```
1 //Source: tutoriel/a_modules/exemples/
      fonctionDupliquee1.ceylon
2 void fonctionDupliquee() {
3   // Fonction dupliquée
4 }
```

Doublon (Seconde déclaration)

```
1 //Source: tutoriel/a_modules/exemples/
      fonctionDupliquee2.ceylon
2 void fonctionDupliquee() {
3   // Fonction dupliquée
4 }
```

Et le résultat :

Doublon (sortie console)

```
1 Duplicate declaration error:
      fonctionDupliquee is declared twice
```

Les déclarations comprennent :

- Les références que l'on pourrait appeler *variable* ou *attribut* dans d'autres langages ;
- Les fonctions qui contrairement aux méthodes n'appartiennent pas à une classe mais à un « package » ;
- Les classes.

Donc contrairement au Java, les « packages » de Ceylon ne contiennent pas uniquement des classes, ce qui est nécessaire, car Ceylon ne dispose pas de membres « static ». Les constantes, les singletons, etc. seront donc définis au niveau des « packages » et non des classes.

Je profite de la présentation des premiers éléments du langage pour attirer l'attention sur deux principes fondateurs de Ceylon. Par défaut, tous les éléments sont « immuables » ; ainsi, les références ne sont pas modifiables et les méthodes ne peuvent être redéfinies. La deuxième chose, c'est que la syntaxe utilise fortement le « sucre syntaxique » de manière intelligente et, nous le verrons, il est tout à fait possible d'écrire la même chose sans ce « sucre ». Dans ce cas, il faudra retourner aux lourdeurs que l'on connaît dans certains langages. Ainsi, il est possible en partie de redéfinir certains opérateurs grâce à ce « sucre syntaxique ».

2.2 Rédaction d'un module

Un module, qu'est-ce que c'est ? Il s'agit d'une bibliothèque avec un nom et une version ; c'est le pendant des bundles en OSGi. Contrairement au fonctionnement natif de Java, Ceylon dipose d'un système de gestion de version des bibliothèques : vous pouvez avoir plusieurs versions d'une même bibliothèque au sein de vos applications sans qu'il n'y ait de conflit.

Commençons donc par créer notre premier module. Dans le répertoire des sources, créez un package tutoriel.modules.demo1 (qui correspond à l'arborescence tutoriel/modules/demo1) et ajoutez le fichier suivant :

module.ceylon

```
1 //Source: tutoriel/a_modules/demo1/
      module.ceylon
2 module tutoriel.modules.demo1 "1.0.0" {}
```

Notez que le nom du module doit être le même que le package racine !

Vous avez désormais un module avec le nom tutoriel.modules.demo1 et la version 1.0.0. Celui-ci contient également un package non partagé du même nom.

Jusqu'à maintenant, nous avons uniquement utilisé le module du langage ceylon.language qui est une dépendance implicite à tous les modules Ceylon. Nous allons désormais utiliser le module ceylon.collection (lien 53) :

Importer le module ceylon.collection

```
1 //Source: tutoriel/a_modules/demo1/
      module.ceylon
2 module tutoriel.modules.demo1 "1.0.0" {
3   shared import ceylon.collection "1.0.0
        ";
4 }
```

Vous pouvez « partager » vos imports (c'est-à-dire les annoter avec shared). Les modules qui importent le vôtre importeront alors automatiquement ceux qui sont partagés. Ceci est particulièrement utile si vous en faites usage dans l'API « publique » de votre module.

Maintenant, nous pouvons créer un module avec une API (partagée) et une implémentation (privée) :

```
1 //Source: tutoriel/a_modules/demo2/
      module.ceylon
2 module tutoriel.modules.demo2 "1.0.0" {
3   import ceylon.collection "1.0.0";
4 }
```

```
1 //Source: tutoriel/a_modules/demo2/
      package.ceylon
2 shared package tutoriel.modules.demo2;
```

```
1 //Source: tutoriel/a_modules/demo2/
      Personne.ceylon
2 import ceylon.collection { MutableList }
3 import tutoriel.modules.demo2.impl {
      PersonneImpl }
4
5 shared interface Personne {
6   shared formal String nom;
7   shared formal String prenom;
8   shared formal MutableList<Personne>
        enfants;
9 }
10
11 shared Personne creer(String nom, String
       prenom) {
12   return PersonneImpl(nom, prenom);
13 }
```

```
1 //Source: tutoriel/a_modules/demo2/impl/
      package.ceylon
2 shared package tutoriel.modules.demo2.
      impl;
```

```
1 //Source: tutoriel/a_modules/demo2/impl/
      PersonneImpl.ceylon
2 import tutoriel.modules.demo2 { Personne
       }
3 import ceylon.collection { MutableList,
      LinkedList }
4
5 shared class PersonneImpl(shared actual
      String nom, shared actual String
      prenom) satisfies Personne {
6   shared actual MutableList<Personne>
        enfants = LinkedList<Personne>();
7   shared actual String string => nom + "
         " + prenom;
8 }
```

On remarquera que pour utiliser des éléments d'un package, il faut utiliser import suivi du nom du package, puis entre la liste des éléments à importer. Contrairement à Java, il n'est pas possible d'utiliser le nom complet au sein du code pour accéder à un élément. Pour gérer les collisions de nom (deux éléments de même nom, mais de packages différents) ou par convenance, il est possible de renommer un élément :

Renommer un import

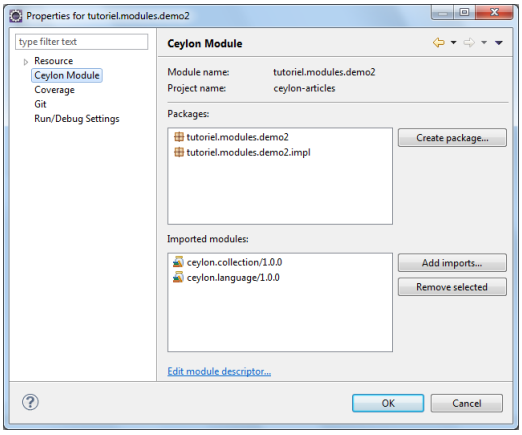
```
1 //Source: tutoriel/a_modules/exemples/
      renommerImport.ceylon
2 import ceylon.collection { List=
      LinkedList }
3
4 void demoRenommageImport() {
5   print(List({"Un", "Deux"}));
6 }
```

Il est également possible de renommer un membre d'un type :

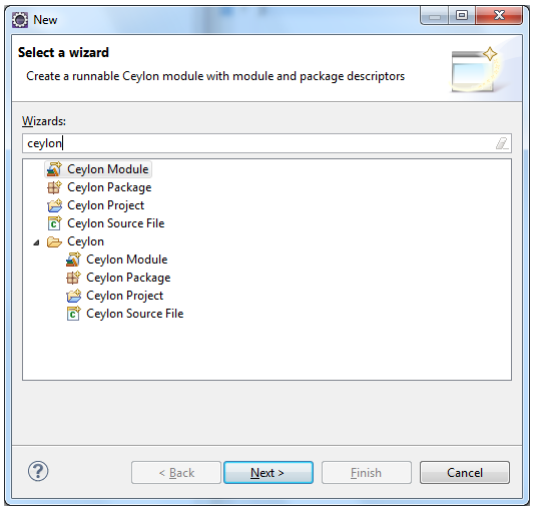
Renommer un membre d'un import

```
1 //Source: tutoriel/a_modules/exemples/
      RenommerImportMembre.ceylon
2 import ceylon.collection { List=
      LinkedList { length=size } }
3
4 void demoRenommageImportMembre() {
5   value list = List({"Un", "Deux"});
6   print("Liste (``list.length``): ``list
        ``");
7 }
```

Vous êtes désormais capable d'écrire vos propres modules. Avant de regarder plus en détail la génération et le contenu d'un module, voici quelques éléments où l'IDE peut vous aider.

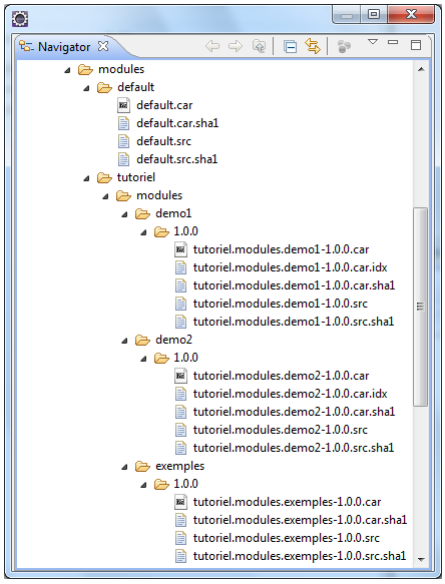


Il existe également plusieurs assistants de création de module, de package et de fichier de sources Ceylon. Si vous êtes dans la perspective « Ceylon », ils sont accessibles depuis le menu contextuel, puis « New » ou depuis le menu « File » -> « New ». Autrement, depuis les mêmes menus, il vous faudra choisir « Other » pour afficher une fenêtre de sélection :



2.3 Génération d'un module

L'avantage d'un IDE, c'est qu'il gère lui-même beaucoup de choses, mais en contrepartie bien des aspects sont masqués. Pour commencer, voyons où trouver ce qu'il a généré. À la racine de votre projet, parcourez le dossier « modules ». Si celui-ci ne s'affiche pas, rafraîchissez votre projet. Vous devriez trouver une arborescence (comme pour les packages) qui correspond aux composantes des noms des modules que vous avez créés. Voici un exemple :



Globalement pour chaque module, on retrouve trois fichiers (ne vous souciez pas des autres) :

— mon.module-1.0.car : il s'agit des binaires de votre module pour Java ;

— mon.module-1.0.js : il s'agit des « binaires » de votre module pour JavaScript ;

— mon.module-1.0.src : il s'agit des sources de votre module pour les éditeurs de code.

Les fichiers « .sha1 » (prononcé « chawane ») sont des fichiers de contrôle qui assurent l'intégrité des fichiers du même nom sans « .sha1 ».

Désormais si vous souhaitez générer la même chose, suivez les étapes suivantes :

— créez un nouveau dossier quelque part sur votre système ;

— créez un sous-répertoire « source » ;

— copiez les fichiers source du module tutoriel.modules.demo2

— ouvrez un interpréteur de commande ;

— placez-vous dans le répertoire parent de « source » ;

— exécutez la commande suivante :

```
1 ceylon compile tutoriel.modules.demo2
```

Si cela ne fonctionne pas, pensez à vérifier que les binaires de Ceylon sont bien dans votre « PATH » et que vous utilisez bien Java en version 7 ou supérieure. Si besoin, vous pouvez spécifier la variable d'environnement JAVA_HOME pour forcer celle à utiliser.

Si tout s'est bien passé, le compilateur vous indique la création du module (avec son nom et sa version). Si vous parcourez l'arborescence, vous découvrirez que le compilateur a créé un répertoire « modules » avec en grande partie les mêmes éléments que vus précédemment.

Mais comment le compilateur a-t-il trouvé le module ceylon.collection ? En réalité, le système modulaire de Ceylon repose sur JBoss Modules (lien 54) (une des briques de base de JBoss AS alias Wildfly). Ceylon localise (à la compilation et à l'exécution) les modules dans des dépôts. Il en existe un certain nombre par défaut dont Ceylon Herd : lien 55. Si vous souhaitez utiliser des dépôts supplémentaires, vous devez utiliser l'option –rep path de la ligne de commande. Si vous voulez exclure les dépôts utilisés par défaut : –no-default-repositories. Vous pouvez retrouver toutes les options ici : lien 56.

Pour finir, si vous souhaitez exécuter vous-mêmes vos programmes, il suffit d'exécuter la commande suivante :

Exécution d'un programme Ceylon

```
ceylon run tutoriel.modules.demo2/1.0.0
```

Néanmoins, rien ne devrait se produire étant donné que nous n'avons pas défini de point d'entrée à notre programme. Pour cela, il suffit de créer une méthode run() à la racine de notre module :

```
//Source: tutoriel/a_modules/demo2/run.ceylon
void run() {
  print("Bonjour depuis le module demo2 !");
  Personne auteur = creer("Mauzaize", "Logan");
  print("Mon créateur est ``auteur``");
}
```

Je vous laisse le soin de recompiler et d'exécuter pour apprécier le résultat.

3 Commentaires

En Ceylon, comme dans beaucoup d'autres langages, pour ajouter des commentaires :

- jusqu'à la fin de la ligne en préfixant par des doubles barres obliques // ;
- sur plusieurs lignes en encadrant avec les caractères /* et */.

Commentaires

```
//Source: tutoriel/b_commentaires/commentaire.ceylon

// Commentaire sur une seule ligne
void commentaire() {
 print("Instruction non commentée"); // Commentaire en fin de ligne
 // print("Instruction commentée");
 /* Commentaire
    sur
    plusieurs
    lignes
  */
}
```

4 Documentation

Bien sûr, ajouter des commentaires à son code permet d'aider à sa compréhension. Mais hélas, ils ne sont pas très exploitables pour documenter vos modules :

Documentation par annotation

```
//Source: tutoriel/c_documentation/a_annotation.ceylon
doc("Ceci est une fonction documentée")
by("Logan")
see(`function documentation`)
void documentation() {
}
```

Ainsi doc, by et see sont des annotations qui sont exploitées par le générateur de documentation de Ceylon.

Il est également possible de documenter une déclaration (i.e. une référence, une fonction ou une classe) simplement en la faisant précéder par une chaîne de caractères :

Documentation avec une chaîne

```
//Source: tutoriel/c_documentation/b_chaine.ceylon
"Une autre façon de documenter une déclaration"
void documentation2() {
}
```

Enfin, il est important de noter que l'on peut formater la documentation à l'aide de la syntaxe Markdown (lien 57) :

Documentation avec Markdown

```
//Source: tutoriel/c_documentation/c_markdown.ceylon
"Ceci est une méthode écrite en [Ceylon][] et documentée avec la syntaxe [Markdown][].

 - - -
 Vous *pourrez* trouver énormément d'informations en lisant la documentation de [Markdown][].

 [Ceylon]: http://ceylon-lang.org/
 [Markdown]: http://daringfireball.net/projects/markdown/syntax"
void documentation3() {
}
```

5 Littéraux

Lorsque l'on démarre avec un langage, on commence souvent par les « littéraux ». Ce sont les valeurs que l'on écrit dans le code source (ex : 12, 3.14, 'a'), car il faut bien commencer par une donnée.

5.1 Booléen

Les booléens sont les types les plus simples que l'on puisse trouver dans un langage. Dans Ceylon, ils sont représentés par le type Boolean et les valeurs true et false. Voici un exemple de booléens :

```
1  //Source: tutoriel/d_litteraux/a_booleen
       .ceylon
2  Boolean vrai = true;
3  Boolean faux = false;
4
5  Boolean ou = true || false;
6  Boolean et = true && false;
7  Boolean non = !true;
```

Attention ! En réalité, true et false ne sont pas des littéraux mais les deux seules instances de la classe Boolean.

— true : lien 58 ;
— false : lien 59 ;
— Boolean : lien 60.

5.2 Nombres

Les nombres sont divisés en deux catégories : les nombres entiers du type Integer (lien 61) et les nombres décimaux du type Float (lien 62). Le code suivant permet de déclarer un entier qui a la valeur 42 et un décimal qui a la valeur 3,14 :

```
1  //Source: tutoriel/d_litteraux/b_nombres
       /a_nombres.ceylon
2  Integer entier = 42;
3  Float pi = 3.14;
```

Vous pouvez bien sûr réaliser différentes opérations sur les nombres :

```
1  //Source: tutoriel/d_litteraux/b_nombres
       /b_operationsEntieres.ceylon
2  void operationsEntieres() {
3    print(1 + 1);    // Addition
4    print(5 - 3);    // Soustraction
5    print(2 * 3);    // Multiplication
6    print(4 / 2 );   // Division entière
7    print(5 / 2 );   //
8    print(5 % 2 );   // Modulo (Reste de la
         division entière)
9    print(2 ^ 3 );   // Élévation à la
       puissance
10 }
```

Comme vous l'aurez remarqué, les opérandes étant entiers, l'opération est dite « entière ». Ainsi 5 / 2 donne 2 et non 2,5. À partir du moment où un opérande est décimal, alors Ceylon utilise les opérations « décimales » :

```
1  //Source: tutoriel/d_litteraux/b_nombres
       /c_operationsDecimales.ceylon
2  void operationsDecimales() {
3    print(1.0 + 1);    // Addition
4    print(5 - 3.0);    // Soustraction
5    print(2.0 * 3);    // Multiplication
6    print(4 / 2.0 );   // Division
7    print(5.0 / 2 );   //
8    print(2 ^ 3.0 );   // Élévation à la
       puissance
9    print(2.0 ^ 3 );   //
10 }
```

On remarquera que l'opération « Modulo » n'est pas disponible pour les décimaux.

5.3 Chaînes de caractères

Les chaînes de caractères sont sûrement les types que l'on manipule le plus dans un programme et chaque langage a ses petites spécificités. Ici, on ne parlera que des littéraux. Une chaîne de caractères est du type String et se déclare simplement entre guillemets (").

```
1  //Source: tutoriel/d_litteraux/c_chaines
       /a_declaration.ceylon
2  String chaine = "Bonjour !";
```

Contrairement à beaucoup d'autres langages, il est possible d'écrire une chaîne sur plusieurs lignes :

```
1  //Source: tutoriel/d_litteraux/c_chaines
       /b_multiligne.ceylon
2  String multiligne = "Première ligne
3                      deuxième ligne";
```

Chaque nouvelle ligne d'une chaîne multiligne doit être au minimum alignée avec la première ligne. Dans le cas contraire, vous aurez une erreur de compilation pour vous le rappeler. Cela permet d'avoir des chaînes bien lisibles !

```
1  //Source: tutoriel/d_litteraux/c_chaines
       /c_multiligneBienLisible.ceylon
2  String multiligneBienLisible = "Racine
3                                  |- A
4                                  |  |- A.
                                      1
5                                  |  |- A.
                                      2
6                                  |- B
7                                  |  |- B.
                                      1";
```

Mais comment écrire une chaîne de caractères qui contient un guillemet ? Pour cela, il faut utiliser une des séquences d'échappements :

1. Le caractère d'échappement \ : \", \\, \n, etc. ;
2. La valeur Unicode : \#0022 ;
3. Le nom Unicode : \QUOTATION MARK.

Ces deux dernières constructions permettent ainsi de traiter les caractères ésotériques non supportés par votre encodage tel que \AIRPLANE

Vous pouvez trouver la liste complète des noms de caractères Unicode depuis cette liste : lien 63

Une dernière possibilité pour écrire une chaîne qui contiendrait des guillemets est d'utiliser des chaînes « verbatim », que l'on traduit par « textuellement ». Comprenez que la chaîne sera interprétée telle quelle. Pour déclarer une chaîne « verbatim », il suffit de la délimiter par trois guillemets :

```
1 //Source: tutoriel/d_litteraux/c_chaines
      /e_verbatim.ceylon
2 String chaineTelleQuelle = """Bonjour, "
      Jean-Pierre""";
```

Pour finir, on a souvent besoin de créer une chaîne de caractères à partir d'une (ou plusieurs) variable(s). En Ceylon, nous allons utiliser des *patrons de chaîne :*

```
1 //Source: tutoriel/d_litteraux/c_chaines
      /f_patron.ceylon
2 String nom = "Ceylon";
3 String patron = "Ceci est un programme é
      crit en ``nom``";
```

Comme dans de nombreux langages de programmation, il est également possible d'utiliser la concaténation :

```
1 //Source: tutoriel/d_litteraux/c_chaines
      /g_concatenation.ceylon
2 String nom = "Ceylon";
3 String patron = "Ceci est un programme é
      crit en " + nom;
```

Cependant, il faut noter une grande différence entre les deux écritures. Les patrons ne tiennent pas compte du type des variables alors que la concaténation ne fonctionne qu'avec des chaînes de caractères. Vous devrez donc convertir les variables en chaînes de caractères avant de les concaténer. Ce qui nous donne :

```
1 //Source: tutoriel/d_litteraux/c_chaines
      /h_patron_vs_concatenation.ceylon
2 String nom = "Ceylon";
3 String prenom = "Logan";
4 Integer age = 28;
5 Integer versionJava = 7;
6 Integer annee = 2013;
7
8 String concatenation2 = "Bonjour, je m'
      appelle " + prenom + " et j'ai " +
      age.string + ". Depuis " + annee.
      string + ", je pratique le langage "
       + nom + " qui nécessite Java " +
      versionJava.string + " au minimum";
9 String patron2        = "Bonjour, je m'
      appelle ``prenom`` et j'ai ``age``.
      Depuis ``annee``, je pratique le
      langage ``nom`` qui nécessite Java
      ``versionJava`` au minimum";
```

On notera que les patrons sont beaucoup plus lisibles, surtout si vous devez utiliser des chaînes multilignes.

5.4 Valeurs et variables

Comme je l'avais évoqué précédemment, Ceylon suppose que les choses sont immuables. Il distingue donc les variables et les valeurs. Par défaut lorsque vous déclarez une donnée, c'est une valeur (ou constante) et il n'est pas autorisé que vous la changiez :

```
1 //Source: tutoriel/d_litteraux/
      d_valeur_variable/a_valeur.ceylon
2 void declarationValeur() {
3   Integer valeur = 1;
4   valeur = 2; // Erreur
5 }
```

Une valeur peut également être initialisée plus tard, mais elle ne pourra l'être qu'une seule fois :

```
1 //Source: tutoriel/d_litteraux/
      d_valeur_variable/
      b_initialisationTardive.ceylon
2 void declarationValeur2() {
3   Integer valeur;
4   valeur = 2; // Ok
5   valeur = 3; // Erreur
6 }
```

Pour qu'une donnée puisse varier, il faut « annoter » (c'est-à-dire ajouter une précision) avec le terme variable :

```
1 //Source: tutoriel/d_litteraux/
      d_valeur_variable/c_variable.ceylon
2 void declarationVariable() {
3   variable Integer var = 1;
4   var = 2;
5 }
```

Dans Ceylon, de nombreux « mots clés » n'en sont en fait pas, mais sont des annotations. C'est le cas de variable mais aussi de shared, abstract, etc. que nous verrons plus tard.

6 Structures de contrôles

6.1 if-else

L'instruction if permet de conditionner l'exécution d'un bloc d'instructions :

```
1 //Source: tutoriel/
      e_structuresDeControles/a_if.ceylon
2 void demoIf() {
3   Integer a = 4;
```

```
4   Integer b = 2;
5
6   if (b != 0) {    // Si b est différent
        de 0, on peut faire la division
7     print("a/b=``a / b``");
8   }
9 }
```

Et else (qui est nécessairement associé à un if) permet de définir une alternative. On peut également les enchaîner :

```
1 //Source: tutoriel/
      e_structuresDeControles/b_if_else.
      ceylon
2 void demoIfElse() {
3   Integer a = 4;
4   Integer b = 2;
5
6   if (a == b) {
7     print("a et b sont égaux");
8   } else if (a > b) {
9     print("a est plus grand que b");
10  } else {
11    print("b est plus grand que a");
12  }
13 }
```

Attention, contrairement à de nombreux langages, les accolades sont obligatoires ; il n'y a pas de bloc implicite. Et cela est vrai pour toutes les structures de contrôles.

6.2 then-else

Contrairement au couple if-else, les couples then-else ne reposent pas sur des blocs, mais des valeurs. Il n'y a donc pas d'accolades et on ne peut pas exécuter d'instruction (sauf appeler une méthode). Ainsi ils permettent moins de choses, mais sont plus concis :

```
1 //Source: tutoriel/
      e_structuresDeControles/c_then_else.
      ceylon
2 void demoThenElse() {
3   Integer a = 4;
4   Integer b = 2;
5
6   String message = (a == b then "a et b
        sont égaux")
7                    else (a >  b then "a est
                         plus grand que b")
8                    else                "b est
                         plus grand que a";
9
10  print(message);
11 }
```

6.3 switch-case

L'instruction « switch » ressemble à ce que l'on peut trouver dans les autres langages :

```
1 //Source: tutoriel/
      e_structuresDeControles/d_switch.
      ceylon
2 void demoSwitch() {
3   Integer a = 0;
4
5   switch (a)
6     case (10) { print("10"); }
7     else { print("Autre chose"); }
8 }
```

Les particularités sont :

1. L'absence d'accolade pour le switch. Le bloc est délimité par les case successifs, optionnellement terminés par un else ;
2. Les tests de chaque case doivent se trouver entre parenthèses ;
3. Chaque case dispose de son propre bloc (comme pour le if, les accolades sont obligatoires). Il n'est donc pas possible d'enchaîner des instructions de deux cas différents comme on pourrait le faire en omettant le break dans d'autres langages ;
4. Dans les autres langages, on utilise généralement le mot clé default pour désigner le cas qui ne correspond à aucun cas énuméré précédemment. Pour rester logique, Ceylon utilise le mot clé else comme pour toutes les autres alternatives ;
5. Ceylon vous oblige à gérer tous les cas, ce qui permet d'identifier rapidement les instructions switch lorsque vous ajoutez un nouveau membre à une énumération ;
6. L'instruction s'utilise avec des expressions de type Integer, Character, String ou les types énumérés (que nous présenterons lors d'un prochain article).

6.4 assert

Une assertion se présente comme un if excepté qu'il n'est pas associé à un bloc d'instructions et lance une exception si au moins une des conditions n'est pas vérifiée :

```
1 //Source: tutoriel/
      e_structuresDeControles/e_assert.
      ceylon
2 void demoAssert() {
3   Boolean estValide = false;
4   assert (estValide);
5   print("Ok, tout va bien");
6 }
```

```
1 ceylon run: Assertion failed
2  violated estValide
```

Il est également possible de définir un message en utilisant une annotation doc sur l'assertion :

```
1 //Source: tutoriel/
      e_structuresDeControles/
      f_assertAvecMessage.ceylon
2 void demoAssertAvecMessage() {
3   Boolean estValide = false;
4   "Non cela ne va pas bien"
5   assert (estValide);
6   print("Ok, tout va bien");
7 }
```

```
ceylon run: Assertion failed: Non cela ne va pas bien
 violated estValide
```

Contrairement à Java, il n'est pas possible de désactiver les assertions.

6.5 while

L'instruction while permet de répéter un bloc de code tant qu'une condition est vérifiée :

```
//Source: tutoriel/e_structuresDeControles/g_while.ceylon
void demoWhile() {
  variable Integer a = 0;
  while (a < 10) {
    print(a);
    a = a + 1;
  }
}
```

Contrairement à ce que l'on peut trouver dans les langages inspirés du C, il n'existe pas d'instructions « do-while ».

6.6 for

Les boucles for permettent l'itération sur des « séquences » d'éléments :

```
//Source: tutoriel/e_structuresDeControles/h_for.ceylon
void demoForEach() {
  Integer[] sequence = [ 0, 1, 2, 3 ];
  for (Integer i in sequence) {
    print(i);
  }
}
```

La boucle for ne permet pas le triptyque « déclaration-condition-incrémentation ». Exemple :

```
for (int i = 0; i <= 3; i++) {
  System.out.println(i);
}
```

À la place, Ceylon utilise des séquences spéciales appelées « plage de valeurs » (« spanned range ») :

```
//Source: tutoriel/e_structuresDeControles/i_for_plage_valeurs.ceylon
void demoFor0a3() {
  for (Integer i in 0..3) {
    print(i);
  }
}
```

Il est également possible de définir une plage inversée :

```
//Source: tutoriel/e_structuresDeControles/j_for_plage_decrementale.ceylon
void demoFor3a0() {
  for (Integer i in 3..0) {
    print(i);
  }
}
```

Alternativement, il est possible de définir une « plage en longueur » (« segmented range »), c'est-à-dire, avec un élément de départ et une longueur de séquence :

```
//Source: tutoriel/e_structuresDeControles/k_for_plage_longueur.ceylon
void demoForDe0Sur4() {
  for (Integer i in 0:4) {
    print(i);
  }
}
```

À noter qu'il n'est pas possible de définir de plage en longueur de manière décrémentale.

Un dernier point important est la possibilité d'exécuter un bloc de code en cas de sortie « normale » de la boucle (ni return, ni break) en utilisant l'instruction else :

```
//Source: tutoriel/e_structuresDeControles/l_for_else.ceylon
void demoForElse() {
  Integer cherche = 2;
  Boolean trouve;
  for (Integer i in 0..1) {
    if (cherche == i) {
      trouve = true;
      break;
    }
  } else {
    trouve = false;
  }
  print(trouve);
}
```

6.7 try-catch-finally

La gestion des exceptions s'effectue avec l'habituel triplet try-catch-finally :

```
//Source: tutoriel/e_structuresDeControles/m_try_catch_finally.ceylon
void demoTryCatchFinally() {
  try {
    assert(1==0);
  } catch (AssertionException|
      NegativeNumberException e) {
    print(e.message);
  } finally {
    print("finally !");
  }
}
```

On remarquera que l'on peut utiliser une syntaxe similaire à celle du « multi-catch » apparue en Java 7. Il est également possible d'utiliser la syntaxe « try-with-resource » :

```
//Source: tutoriel/e_structuresDeControles/n_try_with_resource.ceylon
try (File.Reader reader = file.Reader()) {
  // ...
}
```

Enfin pour capturer toutes les exceptions JavaScript ou les sous-classes de java.lang.Exception, il faudra utiliser l'exception ceylon.language.Exception. Cependant, il n'est pas possible de capturer les autres java.lang.Throwable (et notamment les java.lang.Error).

Dans ce cas, il est possible d'omettre le type de l'exception dans la clause catch :

```
1  //Source: tutoriel/
       e_structuresDeControles/
       o_try_catch_type_implicite.ceylon
2  void demoTryCatchAll() {
3    try {
4      // Code à risque
5    } catch (e) {
6      e.printStackTrace();
7    }
8  }
```

Vous remarquerez que, dans les codes donnés en exemple, les exceptions capturées ne sont pas nécessairement levées par le bloc protégé. Contrairement à Java, Ceylon n'a pas de notion de « checked-exception ». Il n'y a aucune obligation de capturer ou de déclarer une exception qui pourrait se produire. En revanche, je vous conseille fortement de documenter les exceptions levées à l'aide de l'annotation throws.

6.8 Liste de conditions

Toutes les instructions qui reposent sur des conditions (if, assert, while) n'acceptent en réalité pas une unique condition mais une liste :

```
1  //Source: tutoriel/
       e_structuresDeControles/
       p_liste_conditions.ceylon
2  void demoListeDeConditions() {
3    Integer a = 0;
4    Integer b = 1;
5    if (a == 0, b == 1) {
6      print("ok !");
7    } else {
8      print("Tant pis ...");
9    }
10 }
```

Chaque condition de la liste doit être vérifiée, comme si les conditions étaient connectées par un « et » logique. La différence avec une conjonction (enchaînement de « et » logiques), c'est que l'on peut utiliser la « conversion structurée de type » (« structured typecasting »).

Ce terme « barbare » désigne simplement le fait que si vous testez le type ou la non-nullité d'une référence, Ceylon prendra en compte le résultat de la conversion dans les instructions suivantes (y compris les conditions suivantes dans la liste). Voici un exemple :

```
1  //Source: tutoriel/
       e_structuresDeControles/
       q_liste_conditions_conversion_struc-
2  ture.ceylon
3  void demoListeDeConditionsEtConversion-
4  Structure() {
5    Number a = 12;
6    if (is Integer a, a.remainder(5) == 2)
         {
7      print("ok !");
8    } else {
9      print("désolé ...");
10   }
11 }
```

7 Types

7.1 Classe

En Ceylon, comme en Java, les classes se déclarent à l'aide du mot clé class :

```
1  //Source: tutoriel/f_types/a_classe/
       a_declaration.ceylon
2  class MaPremiereClasse() {}
3  MaPremiereClasse uneVariable =
       MaPremiereClasse();
```

La première chose qui saute aux yeux, c'est l'absence de mot clé new. Pour instancier un objet, il suffit d'appeler une classe comme s'il s'agissait d'une fonction. La seconde chose qui frappe est la présence de parenthèses après le nom de classe. En Ceylon, il n'y a pas de « constructeur ». Tout ce que vous allez spécifier dans le corps de votre classe va être utilisé pour construire les instances :

```
1  //Source: tutoriel/f_types/a_classe/
       b_constructeur.ceylon
2  String chaine = "Test";
3  class Constructeur() {
4    String monPremierAttribut;
5    if (chaine.size > 2) {
6      monPremierAttribut = chaine;
7    } else {
8      monPremierAttribut = "Trop court";
9    }
10 }
```

Du fait que le corps de votre constructeur va mélanger des instructions et des déclarations, je vous conseille fortement de regrouper l'initialisation de vos variables par groupes et de bien séparer la déclaration de vos méthodes. N'hésitez pas à utiliser les commentaires pour délimiter graphiquement vos groupes.

Bien entendu, il est possible de passer des paramètres. Mais en plus, on pourra les utiliser comme

des attributs ; cela correspond exactement au comportement des fermetures (lien 64) (closure) :

```
//Source: tutoriel/f_types/a_classe/
    c_constructeur_parametre.ceylon
class ConstructeurAvecParametre(shared
    String unAttribut) {
  shared void afficher() {
    print(unAttribut);
  }
}
void testerConstructeurAvecParametre() {
  ConstructeurAvecParametre a =
      ConstructeurAvecParametre("Un
      attribut");
  print(a.unAttribut);
  a.afficher();
}
```

En Ceylon, il n'est pas utile de créer des méthodes getXXX/setXXX. En effet, il est possible de déclarer des accesseurs (getter) et des mutateurs (setter) qui pourront être utilisés comme des attributs. Commençons par le getter, il suffit de le déclarer comme un attribut sauf que l'on va y associer un bloc de code. Pour le setter, c'est un peu la même chose sauf que l'on remplace le type et les annotations par le mot clé assign et l'on accède à la valeur à définir en utilisant simplement le nom de la « propriété ». Voici un exemple :

```
//Source: tutoriel/f_types/a_classe/
    d_accesseur_mutateur.ceylon
class Personne(shared variable String
    prenom, shared variable String
    nomFamille) {
  shared String nom { return prenom + "
      " + nomFamille; }
  assign nom {
    value prenomEtNom = nom.split().
        iterator();
    if (is String p = prenomEtNom.next()
        ) {
      prenom = p;
    }
    if (is String n = prenomEtNom.next()
        ) {
      nomFamille = n;
    }
  }
}
```

Maintenant que l'on a vu comment créer nos premières classes, nous allons voir comment créer une hiérarchie. Comme Java, Ceylon ne supporte pas l'héritage multiple, mais permet de satisfaire plusieurs interfaces (qui seront abordées dans la section suivante) :

```
//Source: tutoriel/f_types/a_classe/
    e_heritage.ceylon
class Mere() {}
class Fille() extends Mere() {}
```

Il est bien sûr possible de déclarer une classe abstraite en utilisant l'annotation abstract. Pour les méthodes, ce sera avec l'annotation formal (théorique). Lorsque les méthodes seront définies dans une classe descendante, il faudra l'annoter avec actual (concret) ; c'est la même idée que l'annotation Override de Java ou le mot clé override de C#.

```
//Source: tutoriel/f_types/a_classe/
    f_abstraction.ceylon
abstract class Abstraite() {
  shared formal void uneMethode();
}
class Concrete() extends Abstraite() {
  shared actual void uneMethode() {
    print("Méthode concrète !");
  }
}
```

Précédemment, nous avons vu que, par défaut, les éléments sont immuables. Ainsi, une méthode ne peut pas être redéfinie. Pour rendre cela possible, il faut annoter la méthode avec default. La méthode redéfinie devra encore une fois être annotée avec actual :

```
//Source: tutoriel/f_types/a_classe/
    g_redefinition.ceylon
class DefinirMethode() {
  shared default void uneMethode() {
    print("DefinirMethode");
  }
}
class RedefinirMethode() extends
    DefinirMethode() {
  shared actual void uneMethode() {
    print("RedefinirMethode");
  }
}
```

Il est également possible d'utiliser une construction raccourcie pour définir ou redéfinir des méthodes/accesseurs/mutateurs d'une seule instruction. Pour cela, il faut utiliser => à la place des accolades. Dans ce cas, il n'est pas possible de déclarer d'annotation ou de raffiner le type de retour :

```
//Source: tutoriel/f_types/a_classe/
    h_redefinition_rapide.ceylon
class RedefinirRapidementUneMethode()
    extends DefinirMethode() {
  uneMethode() => print("
      RedefinirRapidementUneMethode");
}
```

Un point que nous n'avons pas encore abordé : les classes membres. C'est l'encapsulation d'un type dans un autre ou « nested class » en Java. Une des améliorations qu'apporte Ceylon, c'est la possibilité de raffiner ces classes dans les classes filles. L'intérêt alors, c'est que l'on ne crée pas de nouvelles classes et du point de vue de l'utilisateur de la classe, cela reste abstrait : il se contente de créer une instance de la classe sans se soucier de la classe réelle :

```
//Source: tutoriel/f_types/a_classe/
    i_classe_membre.ceylon
"Abstraction d'une ressource : fichier,
    url, etc."
abstract class Ressource() {
  "Permet de lire le contenu de la
      ressource"
  shared formal class Lecteur() {
    shared formal String lire();
  }
}
```

```
"Représente un contenu sur disque"
class Fichier(shared String chemin)
    extends Ressource() {
  "Gestionnaire de contenu spécifique au
      fichier"
  shared actual class Lecteur() extends
      super.Lecteur() {
    shared actual String lire() => "
        Contenu d'un fichier";
  }
}
"Représente un contenu en mémoire"
class ZoneMemoire() extends Ressource()
    {
  "Gestionnaire de contenu spécifique au
      contenu en mémoire"
  shared actual class Lecteur() extends
      super.Lecteur() {
    shared actual String lire() => "
        Contenu d'une zone mémoire";
  }
}
void demoRessource() {
  Ressource fichier = Fichier("/home/
      ceylon/fichier.txt");
  Ressource memoire = ZoneMemoire();

  print("Fichier : " + fichier.Lecteur()
      .lire());
  print("Memoire : " + memoire.Lecteur()
      .lire());
}
```

```
Fichier : Contenu d'un fichier
Memoire : Contenu d'une zone mémoire
```

7.2 Interface

Contrairement à Java (inférieur à 8), les interfaces de Ceylon peuvent contenir des méthodes concrètes mais aussi des accesseurs et des mutateurs ! Il y a cependant deux limitations : une interface ne peut pas définir d'attributs ou de constructeur. Les interfaces se déclarent avec le mot-clé interface. Pour raffiner une interface, qu'il s'agisse d'une classe ou d'une autre interface, il faut utiliser le mot clé satisfies (satisfaire). S'il y a en plusieurs, il faudra les séparer par un & :

```
//Source: tutoriel/f_types/b_interface/
    a_interfaces.ceylon
interface Vehicule {
  shared variable formal String
      immatriculation;
}

interface Deplacable {
  shared variable formal Integer
      position;

  shared default void deplacer(Integer
      vitesse) {
    position += vitesse;
  }
}

class Voiture("Implémentation par un
    attribut" shared variable actual
    String immatriculation) satisfies
    Vehicule & Deplacable {
  "Implémentation par un accesseur et un
      mutateur"
  shared actual Integer position => 0;
  assign position {
  }
}
```

Si l'utilisation des mixin permet enfin d'hériter de comportements définis de manière générique, elle soulève toujours le problème de l'ambiguïté des noms. Si le cas se présente, le compilateur Ceylon vous avertira par une erreur précisant que votre classe ne peut pas hériter de deux déclarations qui n'ont pas un type parent en commun. Le compilateur vous indiquera également le nom de la déclaration et les interfaces concernées.

Code ambigu

```
//Source: tutoriel/f_types/b_interface/
    b_ambigue.ceylon
interface A {
  shared formal String nom;
}
interface B {
  shared formal String nom;
}
class AB() satisfies A&B {
  shared actual String nom = "toto";
}
```

Code non ambigu

```
//Source: tutoriel/f_types/b_interface/
    c_non_ambigue.ceylon
interface C {
  shared formal String foobar;
}
interface D satisfies C {
  shared actual default String foobar =>
      "D";
}
interface E satisfies C {
  shared actual default String foobar =>
      "E";
}
class CDE() satisfies D&E {
  shared actual String foobar => (super
      of D).foobar + (super of E).foobar
      ;
}
```

On remarquera l'utilisation de la construction (super of) qui permet d'appeler le code défini dans un type parent.

7.3 Objet

En Ceylon, pour déclarer anonymement des classes, on utilise le mot clé object. Si l'objet est défini au niveau du package, il formera un singleton. Dans un autre contexte, une classe ou une méthode (y compris un accesseur ou un mutateur), un nouvel objet sera créé respectivement à chaque instanciation ou appel.

```
1  //Source: tutoriel/f_types/c_objet/
       a_objet.ceylon
2  interface UneClasseAConstruire {
3    shared formal void executer();
4  }
5
6  interface MaFabrique {
7    shared formal UneClasseAConstruire
         construire(String s);
8  }
9
10 object createur satisfies MaFabrique {
11   shared actual UneClasseAConstruire
         construire(String s) {
12     object construction satisfies
           UneClasseAConstruire {
13       shared actual void executer() {
14         print("Une construction(``s``)")
               ;
15       }
16     }
17     return construction;
18   }
19 }
20
21 void demoCreateur() {
22   UneClasseAConstruire a = createur.
         construire("a");
23   UneClasseAConstruire b = createur.
         construire("b");
24   a.executer();
25   b.executer();
26 }
```

Dans le code ci-dessus, on remarque que l'objet construction a « capturé » le paramètre s de son contexte, comme une fermeture (closure).

Les objets peuvent également être utilisés pour raffiner un attribut :

```
1  //Source: tutoriel/f_types/c_objet/
       b_attribut_raffine.ceylon
2  interface Met { /* ... */ }
3  abstract class Repas() {
4    shared formal Met entree;
5    shared formal Met plat;
6    shared formal Met dessert;
7  }
8  class MonRepas() extends Repas() {
9    shared actual object entree satisfies
         Met { /* ... */ }
10   shared actual object plat satisfies
         Met { /* ... */ }
11   shared actual object dessert satisfies
         Met { /* ... */ }
12 }
```

8 Conclusion

Ceci conclut notre tour des principes de base du langage. Désormais, vous devriez avoir une bonne idée de comment coder une application telle que vous le feriez avec votre langage habituel (ex : Java). La partie suivante va se consacrer à ce qui fait l'une des forces de Ceylon et donc ce qui justifierait de l'employer dans vos projets : son système de typage (« Type system »).

Retrouvez l'article de ***Logan Mauzaize*** *en ligne : lien 65*

Tutoriel pour s'authentifier avec Facebook, Twitter ou Google, à l'aide de la bibliothèque PAC4j, en 5 minutes

Les sites Web modernes permettent de s'authentifier via des services comme Facebook ou Gmail en plus du classique formulaire maison. Mais multiplier les solutions, c'est aussi se compliquer la vie à mettre en place de nombreux protocoles. La bibliothèque PAC4j existe heureusement. Elle propose une interface simple pour s'identifier auprès des fournisseurs les plus populaires, en quelques lignes de code seulement. Dans cet article, nous allons voir comment faire cela en cinq minutes chrono.

1 Introduction

Sur un site web, les utilisateurs ont généralement besoin de s'authentifier. Plutôt que d'utiliser une authentification locale propre au site, les utilisateurs souhaitent de plus en plus utiliser leur compte Facebook, Twitter ou Google...

La délégation d'authentification est devenue une problématique récurrente. Pourtant, chaque fournisseur d'identité (protocole OAuth pour Facebook, OpenId pour Google, etc.) utilise des solutions et des bibliothèques différentes, plus ou moins complexes, obligeant les développeurs à tout réapprendre tout le temps.

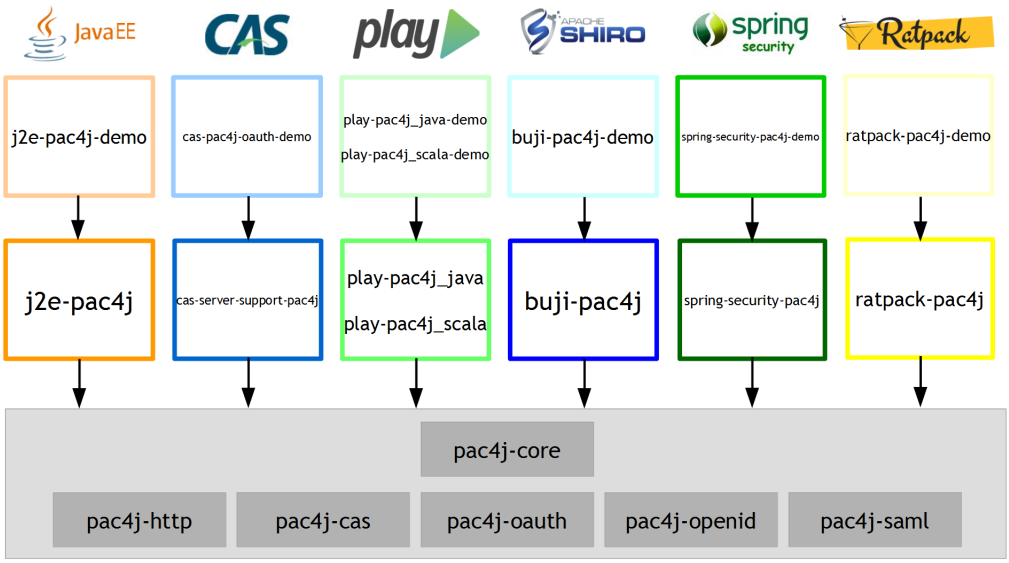
La bibliothèque PAC4j résout ce problème en proposant une solution simple et cohérente pour tous les fournisseurs d'identité, protocoles et frameworks de la Machine virtuelle Java (JVM).

Cet écosystème se compose :

— d'un ensemble de modules (pac4j-core, pac4j-oauth, pac4j-openid...) implémentant les algorithmes et les comportements des différents protocoles et fournisseurs d'identité : le développeur devra choisir les modules adéquats pour le(s) fournisseur(s) d'identité souhaité(s) ;

— d'un ensemble d'implémentations dédiées à différents frameworks de la JVM : j2e-pac4j pour une simple application JEE, spring-security-pac4j pour une application Spring-Security, buji-pac4j pour une application Shiro, play-pac4j pour le framework Play... Le développeur devra choisir la bibliothèque adaptée à son application ;

— d'un ensemble d'applications Web de démonstration (pour JEE, Spring-Security, Shiro, Play, etc.) démontrant les principaux cas d'utilisation.



Pour le moment, PAC4J supporte cinq protocoles et dix-huit fournisseurs d'identité :

— OAuth : Facebook, GitHub, Google, LinkedIn, Twitter, Yahoo, Windows Live, WordPress, DropBox, PayPal, Vk.com, Foursquare, un wrapper OAuth pour le serveur CAS ;

— CAS : serveurs CAS ;

— OpenID : Google ;

— SAML : IdP SAML 2 ;

— HTTP : authentification par formulaire ou par basic auth.

2 Protégez votre application JEE

La première étape pour intégrer l'authentification Facebook dans une application J2E est d'ajouter des dépendances :

— à pac4j-oauth (protocole utilisé par Facebook) ;

— à j2e-pac4j (puisque c'est une application J2E).

Pour Maven par exemple, cela revient à ajouter les dépendances suivantes dans le fichier pom.xml :

```
1
2  <dependency>
3      <groupId>org.pac4j</groupId>
4      <artifactId>j2e-pac4j</artifactId>
5      <version>1.0.2</version>
6  </dependency>
7  <dependency>
8      <groupId>org.pac4j</groupId>
9      <artifactId>pac4j-oauth</artifactId>
10     <version>1.5.0</version>
11 </dependency>
```

La deuxième étape consiste à définir, pour chaque fournisseur d'identité, le client adéquat permettant de « jouer » l'authentification. Pour Facebook, Twitter et Google (fournisseurs OAuth), il faut créer une application OAuth auprès de chacun de ces fournisseurs pour pouvoir s'authentifier. Pour chaque application OAuth, une clé et un « secret » sont fournis par le fournisseur d'identité. Pour cela, on créera une classe dédiée :

```
1
2  public class MyClientsFactory implements
       ClientsFactory {
```

```
3       @Override
4       public Clients build() {
5           final FacebookClient
                facebookClient = new
                FacebookClient("fbKey", "
                fbSecret");
6           final TwitterClient
                twitterClient = new
                TwitterClient("twKey", "
                twSecret");
7           final Google2Client googleClient
                 = new Google2Client("gooKey
                ", "gooSecret");
8           return new Clients("http://
                localhost:8080/callback",
                facebookClient,
                twitterClient, googleClient)
                ;
9           return clients;
10      }
11  }
```

Dans cet exemple, l'URL « http ://localhost : 8080/callback » est l'URL de callback de l'application utilisée par les fournisseurs d'identité pour terminer le process d'authentification.

Pour protéger les URL commençant par « /facebook » et déclencher l'authentification avec Facebook, il faut définir un filtre « RequiresAuthenticationFilter » dans le fichier « web.xml » :

```
1
2   <filter>
3       <filter-name>FacebookFilter</filter-
            name>
4       <filter-class>org.pac4j.j2e.filter.
            RequiresAuthenticationFilter</
            filter-class>
5       <init-param>
6           <param-name>clientsFactory</
                param-name>
7           <param-value>org.leleuj.
                config.MyClientsFactory</
                param-value>
8       </init-param>
9       <init-param>
10          <param-name>clientName</param
                -name>
11          <param-value>FacebookClient</
                param-value>
12      </init-param>
13  </filter>
14  <filter-mapping>
15      <filter-name>FacebookFilter</filter-
            name>
16      <url-pattern>/facebook/*</url-
            pattern>
17      <dispatcher>REQUEST</dispatcher>
18  </filter-mapping>
```

Dans l'exemple, l'application va intercepter toutes les requêtes dont l'URL commencera par « /facebook/ » et les faire passer par le filtre indiqué. Ce filtre pourra réaliser diverses opérations, comme laisser passer (si l'utilisateur est déjà authentifié) ou rediriger vers une page de connexion.

Pour que le processus d'authentification fonctionne (en définissant l'URL de callback), il est nécessaire de définir le filtre « CallbackFilter » dans le fichier « web.xml » :

```
1
2   <filter>
3       <filter-name>CallbackFilter</filter-
            name>
4       <filter-class>org.pac4j.j2e.filter.
            CallbackFilter</filter-class>
5       <init-param>
6           <param-name>clientsFactory</
                param-name>
7           <param-value>org.leleuj.
                config.MyClientsFactory</
                param-value>
8       </init-param>
9       <init-param>
10          <param-name>defaultUrl</param-
                name>
11          <param-value>/</param-value>
12      </init-param>
13  </filter>
14  <filter-mapping>
15      <filter-name>CallbackFilter</filter-
            name>
16      <url-pattern>/callback</url-pattern>
17      <dispatcher>REQUEST</dispatcher>
18  </filter-mapping>
```

Il faut bien voir que l'utilisateur peut déclencher volontairement ou non l'authentification. Il la déclenche involontairement lorsqu'il tente d'accéder à une partie protégée du site. Il peut aussi la déclencher, volontairement cette fois, lorsqu'il clique sur le bouton/lien « Log in ». D'ailleurs, sur de nombreux sites, comme sur les webmails (Gmail, Hotmail, etc.), c'est souvent la seule action possible.

Si on veut explicitement déclencher l'authentification (pour Facebook par exemple), il suffit de calculer l'URL de redirection appropriée (vers Facebook) :

```
1
2   FacebookClient facebookClient = (
        FacebookClient) clients.findClient("
        FacebookClient");
3   String redirectionUrl = facebookClient.
        getRedirectAction(context, false,
        false).getLocation();
```

3 Protégez votre application Spring-Security

La bibliothèque Spring-Security est certainement l'une des plus populaires pour gérer la sécurité de son site. Dans cette section, nous allons voir comment mixer PAC4j et Spring-Security, car le fonctionnement est un peu spécifique.

La première étape pour intégrer l'authentification Facebook dans une application Spring-Security est là encore d'ajouter les dépendances correspondantes :

— à pac4j-oauth (protocole utilisé par Face-

book) ;

— à spring-security-pac4j (puisque c'est une application Spring Security).

Pour Maven par exemple, il faut donc ajouter les dépendances suivantes dans le fichier « pom.xml » :

```
1
2  <dependency>
3      <groupId>org.pac4j</groupId>
4      <artifactId>spring-security-pac4j</
           artifactId>
5      <version>1.2.2</version>
6  </dependency>
7  <dependency>
8      <groupId>org.pac4j</groupId>
9      <artifactId>pac4j-oauth</artifactId>
10     <version>1.5.0</version>
11 </dependency>
```

Dans une application Spring-Security, les clients sont définis dans le contexte Spring :

```
1
2  <bean id="facebookClient" class="org.pac
       4j.oauth.client.FacebookClient">
3      <property name="key" value="fbKey" /
           >
4      <property name="secret" value="
           fbSecret" />
5  </bean>
6  <bean id="twitterClient" class="org.pac4
       j.oauth.client.TwitterClient">
7      <property name="key" value="twKey" /
           >
8      <property name="secret" value="
           twSecret" />
9  </bean>
10 <bean id="googleClient" class="org.pac4j
       .oauth.client.Google2Client">
11     <property name="key" value="gooKey"
           />
12     <property name="secret" value="
           gooSecret" />
13 </bean>
14 <bean id="clients" class="org.pac4j.core
       .client.Clients">
15     <property name="callbackUrl" value="
           http://localhost:8080/callback"
           />
16     <property name="clients">
17           <list>
18                <ref bean="facebookClient
                      " />
19                <ref bean="twitterClient"
                      />
20                <ref bean="googleClient"
                      />
21           </list>
22     </property>
23 </bean>
```

Pour protéger les URL commençant par « /facebook », une section « security :http » doit être définie avec l'entry point et le filtre adéquats dans le contexte de sécurité Spring :

```
1
2  <security:http entry-point-ref="
       facebookEntryPoint">
3      <security:custom-filter after="
           CAS_FILTER" ref="clientFilter" /
           >
4      <security:intercept-url pattern="/
           facebook/**"     access="
           IS_AUTHENTICATED_FULLY" />
5  </security:http>
6  <bean id="facebookEntryPoint"     class=
       "org.pac4j.springframework.security.
       web.ClientAuthenticationEntryPoint">
7      <property name="client" ref="
           facebookClient" />
8  </bean>
```

Pour que le process d'authentification fonctionne (en définissant l'URL de callback), il est nécessaire de définir le filtre « ClientAuthenticationFilter » et le provider « ClientAuthenticationProvider » dans le contexte de sécurité Spring :

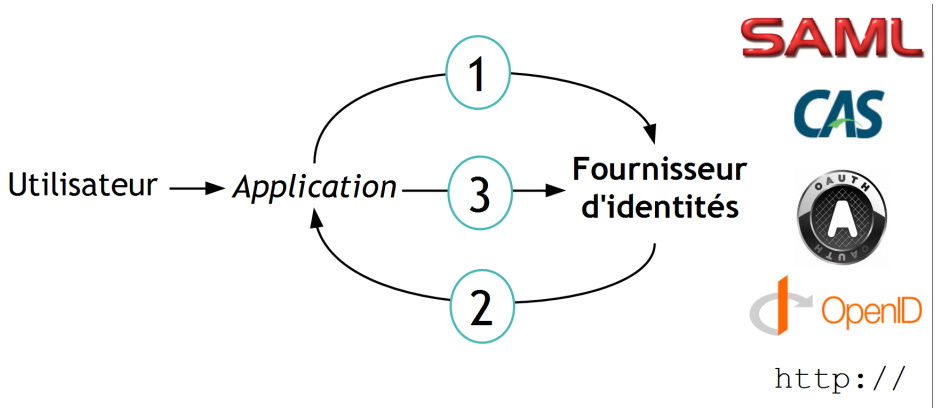
```
1
2  <bean id="clientFilter" class="org.pac4j
       .springframework.security.web.
       ClientAuthenticationFilter">
3      <constructor-arg value="/callback"/>
4      <property name="clients" ref="
           clients" />
5      <property name="
           authenticationManager" ref="
           authenticationManager" />
6  </bean>
7  <bean id="clientProvider" class="org.pac
       4j.springframework.security.
       authentication.
       ClientAuthenticationProvider">
8      <property name="clients" ref="
           clients" />
9  </bean>
```

4 Protégez vos autres applications

Vous pouvez également protéger vos applications Shiro, Play framework ou ratpack de manière similaire à ce qui a été fait pour JEE et Spring-Security en utilisant les bibliothèques dédiées.

Le serveur CAS utilise également la librairie pac4j via le module cas-server-support-pac4j pour déléguer l'authentification.

Si vous souhaitez utiliser un fournisseur d'identité qui n'existe pas encore pour pac4j, vous pouvez le développer vous-même. Ou si vous utilisez un framework Java pour lequel il n'existe aucune implémentation utilisable de pac4j, vous pouvez également la créer vous-même. Tous les protocoles implémentés par pac4j suivent la cinématique suivante :

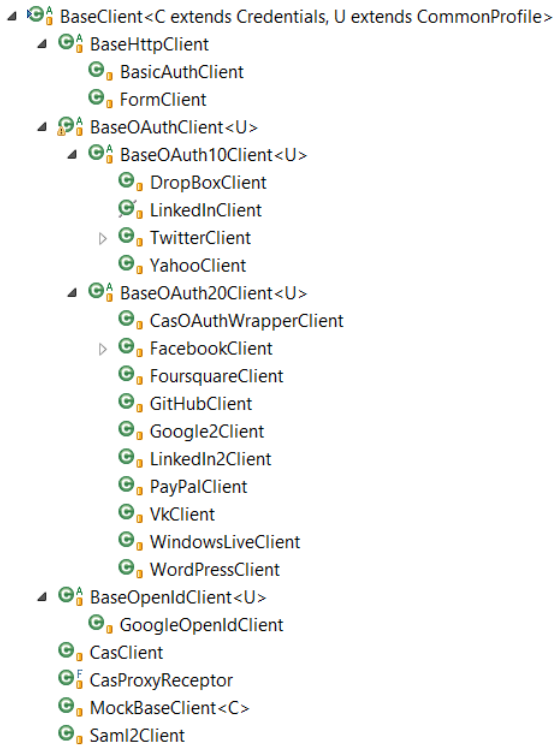


Lorsque l'utilisateur accède à une application protégée, 1) il est redirigé vers le fournisseur d'identité et s'y authentifie, 2) il est redirigé vers l'application avec des informations spécifiques et 3) l'application utilise ces informations spécifiques pour valider l'authentification de l'utilisateur et récupérer son profil.

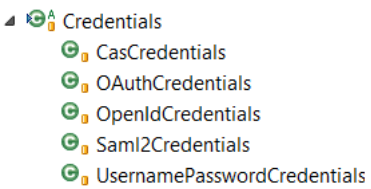
Cette cinématique est représentée par l'interface « Client » qui doit être implémentée par tous les clients :

```
1
2  public interface Client<C extends
       Credentials, U extends UserProfile >
        {
3
4      void redirect(WebContext context,
           boolean protectedTarget, boolean
            ajaxRequest) throws
           RequiresHttpAction;
5
6      C getCredentials(WebContext context)
            throws RequiresHttpAction;
7
8      U getUserProfile(C credentials,
           WebContext context);
9  }
```

Donc tous les clients comme Facebook, Twitter, Google, etc. (définis par les classes FacebookClient, TwitterClient, Google2Client, etc.) implémentent cette interface. En fait, il existe une hiérarchie complète de classes sous l'implémentation mère « org.pac4j.core.client.BaseClient » :

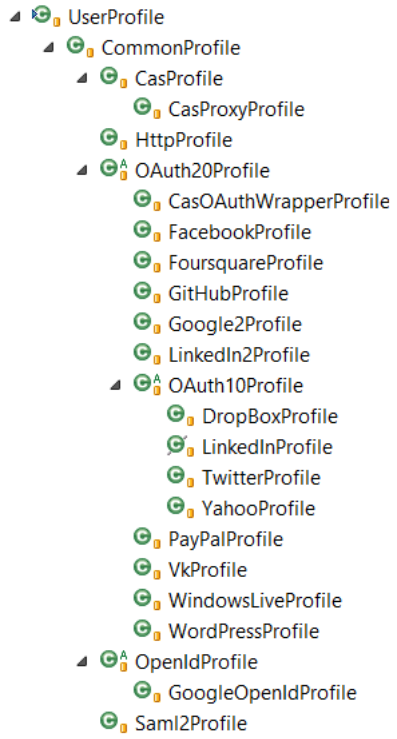


Pour démarrer le processus d'authentification pour Facebook par exemple, la méthode « redirect » du client FacebookClient doit être appelée pour rediriger l'utilisateur vers le fournisseur d'identité afin de l'authentifier. Après une authentification réussie, l'utilisateur est redirigé vers l'URL de callback de l'application sur laquelle la méthode « getCredentials » doit être appelée afin de récupérer les informations spécifiques (« credentials ») renvoyées par le fournisseur d'identité. On a différents types de « credentials » suivant les protocoles :



Avec ces « credentials », on récupère finalement le profil de l'utilisateur authentifié en appelant la mé-

thode « getUserProfile » du client. Là aussi, on a une hiérarchie complète de profils suivant les différents clients :



Tous les profils utilisateurs héritent de la classe « org. pac4j.core.profil.CommonProfile », qui regroupe un ensemble de méthodes habituelles : getEmail, getFirstName, getLastName...

Un profil utilisateur est défini par un identifiant, une liste d'attributs, des rôles, des permissions et un statut « remember-me ». Ces attributs sont récupérés du fournisseur d'identité et peuvent être définis précisément en implémentant l'interface « org.pac4j.core.profile.AttributesDefinition » qui donne les conversions à effectuer pour chacun des attributs récupérés. Par exemple, voici la définition d'attributs pour Facebook :

```
1
2  public class
       FacebookAttributesDefinition extends
        OAuthAttributesDefinition {
3
4      public static final String NAME = "
           name";
5      public static final String
           FIRST_NAME = "first_name";
6      public static final String
           MIDDLE_NAME = "middle_name";
7      public static final String LAST_NAME
            = "last_name";
8      public static final String GENDER =
           "gender";
9      public static final String LOCALE =
           "locale";
10     public static final String LANGUAGES
            = "languages";
11
12     ...
13
```

```
14     public FacebookAttributesDefinition
           () {
15         final String[] names = new
               String[] {
16             NAME, FIRST_NAME,
                   MIDDLE_NAME, LAST_NAME,
                   LINK, USERNAME,
                   THIRD_PARTY_ID,
17             BIO, EMAIL, POLITICAL,
                   QUOTES, RELIGION,
                   WEBSITE };
18
19         for (final String name : names)
               {
20             addAttribute(name,
                   Converters.
                   stringConverter);
21         }
22         addAttribute(TIMEZONE,
               Converters.integerConverter)
               ;
23         addAttribute(VERIFIED,
               Converters.booleanConverter)
               ;
24         addAttribute(GENDER, Converters.
               genderConverter);
25         addAttribute(LOCALE, Converters.
               localeConverter);
26         addAttribute(UPDATED_TIME,
               Converters.dateConverter);
27         addAttribute(BIRTHDAY,
               FacebookConverters.
               birthdayConverter);
28         addAttribute(RELATIONSHIP_STATUS
               , FacebookConverters.
               relationshipStatusConverter)
               ;
29         addAttribute(LANGUAGES,
               FacebookConverters.
               listObjectConverter);
30
31         ...
```

Ainsi, pour un nouveau fournisseur d'identité, il vous faut créer votre client « MonFournisseurClient », gérant des informations spécifiques « MonFournisseurCredentials » et renvoyant un profil utilisateur « MonFournisseurProfile »...

Si aucun fournisseur d'identité ne vous manque, mais que vous n'avez pas d'implémentation PAC4J pour votre framework, vous pouvez créer vous-même votre bibliothèque « monframework-pac4j ».

Cette bibliothèque sera basée sur la dernière version du module « pac4j-core » qui est le socle commun (indépendant des protocoles) de PAC4J. Pour Maven par exemple, cela donnerait l'ajout suivant dans le fichier « pom.xml » :

```
1
2  <dependency>
3      <groupId>org.pac4j</groupId>
4      <artifactId>pac4j-core</artifactId>
5      <version>1.5.0</version>
6  </dependency>
```

Vous devrez utiliser la classe « org.pac4j.core.client. Clients » (notez le « s » à la fin), qui permet de regrouper plusieurs clients sur une même URL de callback (sinon il vous faudrait plusieurs URL de callback : /callbackFacebook, /callbackTwitter, /call-

backGoogle...ce qui complexifierait grandement les choses). La bibliothèque pac4j interagit avec les requêtes/réponses HTTP via l'abstraction/interface « org.pac4j.core.context.WebContext » qui représente le contexte web courant. Une implémentation pour JEE existe via la classe « org.pac4j.core.context.J2EContext » mais vous serez peut-être amené à créer la vôtre, par exemple :

— le contexte « io.buji.pac4j.ShiroWebContext » dédié au framework de securité Shiro ;

— le contexte « org.pac4j.play.java.JavaWebContext » pour le framework Play en langage Java.

Ainsi, en utilisant la classe « Clients », le contexte web approprié et la méthode « redirect » du client choisi, vous pouvez sécuriser une URL et rediriger l'utilisateur vers le fournisseur d'identité pour qu'il s'authentifie.

La sécurisation d'une URL dépend évidemment des possibilités offertes par le framework. Dans le cas de JEE, le filtre « RequiresAuthenticationFilter » assure ce travail :

```
1
2  protected void internalFilter(final
       HttpServletRequest request, final
       HttpServletResponse response, final
       HttpSession session, final
       FilterChain chain) throws
       IOException, ServletException {
3
4      final CommonProfile profile =
           UserUtils.getProfile(request);
5      logger.debug("profile : {}", profile
           );
6
7      // profile not null, already
           authenticated -> access
8      if (profile != null) {
9          chain.doFilter(request, response
               );
10
11     } else {
12         // no authentication tried ->
               redirect to provider
13         // keep the current url
14         String requestedUrl = request.
               getRequestURL().toString();
15         String queryString = request.
               getQueryString();
16         if (CommonHelper.isNotBlank(
               queryString)) {
17             requestedUrl += "?" +
                   queryString;
18         }
19         logger.debug("requestedUrl : {}"
               , requestedUrl);
20         session.setAttribute(
               ORIGINAL_REQUESTED_URL,
               requestedUrl);
21
22         // compute and perform the
               redirection
23         final WebContext context = new J
               2EContext(request, response)
               ;
24         Client<Credentials,
               CommonProfile> client =
               ClientsConfiguration.
               getClients().findClient(this
               .clientName);
25
26         try {
27             client.redirect(context,
                   true, false);
28         } catch (RequiresHttpAction e) {
29             logger.debug("extra HTTP
                   action required : {}", e
                   .getCode());
30         }
31     }
32 }
```

Dans le cas de Spring-Security, c'est l'entry point « ClientAuthenticationEntryPoint » (déclenché lorsqu'une authentification est nécessaire) qui joue ce rôle :

```
1
2  public final class
       ClientAuthenticationEntryPoint
       implements AuthenticationEntryPoint,
        InitializingBean {
3
4      private Client<Credentials,
           UserProfile> client;
5
6      public void commence(final
           HttpServletRequest request,
           final HttpServletResponse
           response, final
           AuthenticationException
           authException) throws
           IOException, ServletException {
7          logger.debug("client : {}", this
               .client);
8          final WebContext context = new J
               2EContext(request, response)
               ;
9
10         try {
11             this.client.redirect(context
                   , true, false);
12         } catch (final
               RequiresHttpAction e) {
13             logger.debug("extra HTTP
                   action required : {}", e
                   .getCode());
14         }
15     }
```

Après une authentification réussie, l'utilisateur est redirigé sur l'URL de callback de l'application où l'implémentation de pac4j adéquate doit récupérer les « credentials » et le profil utilisateur en utilisant les Clients et les méthodes « getCredentials » et « getUserProfile ».

Dans le cas de JEE, c'est le filtre « CallbackFilter » qui effectue ces tâches et initialise ainsi le contexte de sécurité (avec le profil de l'utilisateur) :

```
1
2  public class CallbackFilter extends
       ClientsConfigFilter {
3
4      protected void internalFilter(final
           HttpServletRequest request,
           final HttpServletResponse
           response,
5           final HttpSession session,
                final FilterChain chain)
```

```
            throws IOException,
            ServletException {

        final WebContext context = new J
            2EContext(request, response)
            ;
        final Client client =
            ClientsConfiguration.
            getClients().findClient(
            context);
        logger.debug("client : {}",
            client);

        final Credentials credentials;
        try {
            credentials = client.
                getCredentials(context);
        } catch (final
            RequiresHttpAction e) {
            logger.debug("extra HTTP
                action required : {}", e
                .getCode());
            return;
        }
        logger.debug("credentials : {}",
             credentials);

        // get user profile
        final CommonProfile profile = (
            CommonProfile) client.
            getUserProfile(credentials,
                context);
        logger.debug("profile : {}",
            profile);

        if (profile != null) {
            // only save profile when it
                's not null
            UserUtils.setProfile(session
                , profile);
        }

        final String requestedUrl = (
            String) session.getAttribute
            (
            RequiresAuthenticationFilter
            .ORIGINAL_REQUESTED_URL);
        logger.debug("requestedUrl : {}"
            , requestedUrl);
        if (CommonHelper.isNotBlank(
            requestedUrl)) {
            response.sendRedirect(
                requestedUrl);
        } else {
            response.sendRedirect(this.
                defaultUrl);
        }
    }
}
```

Dans le cas de Spring-Security, la récupération des « credentials » et du profil utilisateur s'effectue en deux temps via le filtre approprié (lien 66) et le provider adéquat (lien 67).

Pour créer votre propre implémentation de pac4j pour votre framework, vous devrez donc fournir les mécanismes nécessaires pour protéger des URL et terminer le processus d'authentification.

5 Conclusion

C'est souvent la croix et la bannière de sécuriser une application Web à l'aide des protocoles à la mode (OAuth, OpenId, etc.) et d'autant plus lorsqu'on en utilise plusieurs. Or les fournisseurs comme Google, Linked'in, Yahoo (et bien d'autres) ne rendent pas cette tâche si facile, malgré leurs efforts.

Simplifiez-vous donc la vie et utilisez pac4j pour gérer toutes vos authentifications sur toutes vos plateformes.

Alors ? A-t-on réussi le pari de la connexion OAuth, OpenId, etc. en moins de cinq minutes ? À condition d'avoir tous les identifiants et les URL sous la main, je pense sincèrement que c'est jouable. Mais la balle est dans votre camp. À vous de nous dire si c'est gagné.

Dans tous les cas, l'équipe qui développe PAC4J est très réactive et répond volontiers aux questions. N'hésitez donc pas à consulter le site de la bibliothèque et, si besoin, à soumettre un ticket.

Retrouvez l'article de ***Jerome Leleu*** *et* ***Thierry-Leriche-Dessirier*** *en ligne : lien 68*

NoSQL

Les derniers tutoriels et articles

Tutoriel d'introduction à Apache Hadoop

Ce premier article introductif s'intéresse à présenter le système de fichiers HDFS (Hadoop Distributed File System) et le modèle de programmation MapReduce.
Mes articles consacrés à Hadoop sont décrits ci-dessous :

- Généralités sur HDFS et MapReduce : lien 69 ;
- Installation et configuration d'un cluster simple nœud avec Cloudera CDH 5 : lien 70 ;
- Développement, test et débogage de programmes MapReduce avec Cloudera CDH 5 ;
- Installation et configuration d'un cluster multinœud avec Cloudera CDH 5.

Je tiens à préciser que je ne suis pas un spécialiste d'Hadoop. Ces articles sont le résultat d'une veille technologique. Ils seront sûrement améliorés au fur et à mesure de mes différentes découvertes et de l'exploitation d'Hadoop lors de cas réels.
L'objectif visé par ces articles est de démystifier Apache Hadoop et de tenter de rendre sa compréhension aussi facile qu'un jeu d'enfant.

1 Généralités

Nous sommes actuellement dans l'ère de la production massive de données. D'une part, les applications génèrent des données issues des logs, des réseaux de capteurs, des rapports de transactions, des traces de GPS, etc. et d'autre part, les individus produisent des données telles que des photographies, des vidéos, des musiques ou encore des données sur l'état de santé (rythme cardiaque, pression ou poids).

Un problème se pose alors quant au stockage et à l'analyse des données. La capacité de stockage des disques durs augmente mais le temps de lecture croît également. Il devient alors nécessaire de paralléliser les traitements en stockant sur plusieurs unités de disques durs. Toutefois, cela soulève forcément le problème de fiabilité des disques durs qui engendre la panne matérielle. La solution envisagée est la duplication des données comme le ferait un système RAID.

Apache Hadoop (High-availability distributed object-oriented platform) est un système distribué qui répond à ces problématiques. D'une part, il propose un système de stockage distribué via son système de fichier HDFS (Hadoop Distributed File System) et ce dernier offre la possibilité de stocker la donnée en la dupliquant, un cluster Hadoop n'a donc pas besoin d'être configuré avec un système RAID qui devient inutile. D'autre part, Hadoop fournit un système d'analyse des données appelé MapReduce. Ce dernier officie sur le système de fichiers HDFS pour réaliser des traitements sur des gros volumes de données.

Hadoop a été créé par Doug Cutting pour les besoins du projet Apache Nutch (lien 71), un moteur de recherche *open source*. À noter que Doug Cutting n'est pas un novice puisqu'il a également créé Apache Lucene (lien 72) la bibliothèque de recherche plein texte. Lorsque le projet Apache Nutch a démarré en 2002, les contributeurs ont compris que l'architecture d'origine ne pourrait pas tenir la montée en charge sur plus de 20 milliards de pages depuis le Web. Google publie en 2003 un article présentant l'architecture de son système de fichiers distribué GFS (Google's distributed filesystem) : lien 73. Google publie ensuite en 2004 un article introduisant le système MapReduce pour l'analyse des données d'un système GFS : lien 74. Doug Cutting a décidé de reprendre les concepts présentés par les deux articles pour résoudre les problèmes rencontrés depuis le projet Apache Nutch. En 2006, Hadoop devient un sous-projet d' Apache Lucene et en 2008 un projet indépendant (lien 75) de la fondation Apache.

Pour la petite histoire, le logo Hadoop est basé sur le doudou d'un des enfants de Doug Cutting.

2 Intérêt et usages

Hadoop n'a d'intérêt que s'il est utilisé dans un environnement composé de plusieurs machines. Utiliser Hadoop dans un environnement monomachine, comme nous allons le faire dans le prochain tutoriel, n'a de sens que pour tester la configuration de l'installation ou fournir un environnement de développement MapReduce (prochain article). Hadoop n'a également pas d'intérêt pour les données de petite taille. C'est même l'effet inverse qui pourrait se produire. Si les données ne dépassent pas les limites de la mémoire ou des disques durs du marché (actuellement la limite est aux alentours de cinq téraoctets), posez-vous la question sur l'utilité d'utiliser Hadoop face à des solutions utilisant des bases de données relationnelles. Vous trouverez sur ce lien une analyse simple mais efficace : lien 76.

Il est à noter que Yahoo (lien 77) a massivement investi dans le projet Hadoop. La société utilise Hadoop pour sa page d'accueil (publicités ciblées, contenus adaptés, le « top recherche » des utilisateurs...) depuis 2006 et fut l'un des plus grands utilisateurs, testeurs et contributeurs du projet. À titre d'exemple, Yahoo a monté un serveur Hadoop de 2000 nœuds afin de trier vingt téraoctets de données. La tâche a pris 2 heures et 30 minutes (lien 78).

Vous trouverez sur cette page (lien 79), la liste des entreprises et institutions publiques qui utilisent Hadoop.

Concernant les usages, ils sont relativement variés. Les propos qui vont suivre sont basés sur ce livre en libre accès : Hadoop Illuminated (lien 80). Je vous invite également à consulter le Hadoop User Group (lien 81) qui contient de très bonnes vidéos sur des retours d'expérience sur Apache Hadoop dans les entreprises françaises.

- **Santé** : des chercheurs d'un hôpital pour enfants à Los Angeles (lien 82) stockent et analysent des données issues des capteurs sensoriels. Apache HDFS est utilisée pour ses capacités de stockage (environ un téraoctet par jour) et sa facilité de scalabilité à moindre coût. L'entreprise NextBio (lien 83) analyse de grandes quantités de données sur des génomes humains (lien 84). L'analyse est réalisée par des *batches* MapReduce et le stockage (plusieurs téraoctets) est effectué par Hbase, la base de données clé/valeur.
- **Télécommunication** : Nokia (lien 85) collecte et analyse de grandes quantités de données (un téraoctet par jour) issues des téléphones portables de sa marque (lien 86).
- **Énergie** : des travaux sont en cours chez EDF (lien 87) pour s'intéresser à la modélisation, l'analyse et la prévision des consommations électriques en captant les informations de compteurs intelligents. Une solution à base d'Hadoop est utilisée pour le stockage et l'analyse de courbes de charge (séries temporelles) (lien 88 et lien 89).
- **Transport** : US Xpress (lien 90), une importante société de transport routier aux États-Unis utilise Apache Hadoop pour stocker les données de capteurs transmises par leur flotte de véhicules (données de géolocalisation, par exemple). L'analyse permet d'optimiser le déplacement de véhicules dans le but d'économiser sur le coût du carburant (lien 91). Une expérimentation (lien 92) a également donné lieu à l'utilisation d'Apache Hadoop pour la résolution du problème du voyageur de commerce dont le but est « étant donné n points (des « villes ») et les distances séparant chaque point, trouver un chemin de longueur totale minimale qui passe exactement une fois par chaque point et revienne au point de départ » (lien 93). Il est intéressant de voir 1) la décomposition du problème en MapReduce et 2) que Hadoop n'a pu proposer des solutions approchées de manière efficace.
- **Vente** : Etsy (lien 94), un site de e-commerce analyse les gros volumes des données de logs pour déterminer le comportement utilisateur ou les recommandations de recherche (lien 95).
- **Images et vidéos** : SkyBox (lien 96), qui fournit un système pour capturer des vidéos et des images satellite n'importe où sur Terre utilise Hadoop pour effectuer des traitements en parallèle sur les images. Le traitement se fait via des algorithmes écrits en C++, les développeurs de la société ont réalisé un framework permettant d'exécuter du code natif à partir du framework MapReduce écrit en Java (lien 97).

3 Hadoop Distributed File System (HDFS)

HDFS (Hadoop Distributed File System) reprend de nombreux concepts proposés par des systèmes de fichiers classiques comme ext2 pour Linux ou FAT pour Windows. Nous retrouvons donc la notion de blocs (la plus petite unité que l'unité de stockage peut gérer), les métadonnées qui permettent de retrouver les blocs à partir d'un nom de fichier, les droits ou encore l'arborescence des répertoires.

Toutefois, HDFS se démarque d'un système de fichiers classique pour les principales raisons suivantes.

— HDFS n'est pas solidaire du noyau du système d'exploitation. Il assure une portabilité et peut être déployé sur différents systèmes d'exploitation. Un des inconvénients est de devoir solliciter une application externe pour monter une unité de disque HDFS.

— HDFS est un système distribué. Sur un système classique, la taille du disque est généralement considérée comme la limite globale d'utilisation. Dans un système distribué comme HDFS, chaque nœud d'un cluster correspond à un sous-ensemble du volume global de données du cluster. Pour augmenter ce volume global, il suffira d'ajouter de nouveaux nœuds. On retrouvera également dans HDFS, un service central appelé Namenode qui aura la tâche de gérer les métadonnées.

— HDFS utilise des tailles de blocs largement supérieures à ceux des systèmes classiques. Par défaut, la taille est fixée à 64 Mo. Il est toutefois possible de monter à 128 Mo, 256 Mo, 512 Mo voire 1 Go. Alors que sur des systèmes classiques, la taille est généralement de 4 Ko, l'intérêt de fournir des tailles plus grandes permet de réduire le temps d'accès à un bloc. Notez que si la taille du fichier est inférieure à la taille d'un bloc, le fichier n'occupera pas la taille totale de ce bloc.

— HDFS fournit un système de réplication des blocs dont le nombre de réplications est configurable. Pendant la phase d'écriture, chaque bloc correspondant au fichier est répliqué sur plusieurs nœuds. Pour la phase de lecture, si un bloc est indisponible sur un nœud, des copies de ce bloc seront disponibles sur d'autres nœuds.

3.1 Namenode

Un Namenode est un service central (généralement appelé aussi maître) qui s'occupe de gérer l'état du système de fichiers. Il maintient l'arborescence du système de fichiers et les métadonnées de l'ensemble des fichiers et répertoires d'un système Hadoop. Le Namenode a une connaissance des Datanodes (étudiés juste après) dans lesquels les blocs sont stockés. Ainsi, quand un client sollicite Hadoop pour récupérer un fichier, c'est via le Namenode que l'information est extraite. Ce Namenode va indiquer au client quels sont les Datanodes qui contiennent les blocs. Il ne reste plus au client qu'à récupérer les blocs souhaités.

Toutes ces métadonnées, hormis la position des blocs dans les Datanodes, sont stockées physiquement sur le disque système dans deux types de fichiers spécifiques edits_xxx et fsimage_xxx.

La connaissance de la position des blocs dans les Datanodes est reconstruite à chaque démarrage du Namenode dans un mode appelé safe mode. Pendant le safe mode, l'écriture sur HDFS est impossible, le Namenode charge les fichiers edits_xxx et fsimage_xxx et attend le retour des Datanodes sur la position des blocs. Une fois toutes les opérations réalisées, le safe mode est relâché et l'accès en écriture est de nouveau autorisé. Soyez patient sur la durée du safe mode. Celui-ci peut être très long si vous avez beaucoup de fichiers à traiter.

Comme vous l'aurez remarqué, le Namenode charge tout en mémoire. Cela devient donc problématique si vous avez énormément de petits fichiers à gérer. D'après la documentation officielle de Cloudera (lien 98), chaque fichier, répertoire et bloc dans HDFS est représenté comme un objet dans la mémoire et occupe 150 octets. Si, par exemple, vous avez 10 millions de fichiers à gérer, le Namenode devra disposer d'un minimum de 1,5 Go de mémoire. C'est donc un point important à prendre en compte lors du dimensionnement de votre cluster. Le Namenode est relativement gourmand en mémoire.

3.2 Secondary Namenode

Le Namenode dans l'architecture Hadoop est un point unique de défaillance (Single Point of Failure en anglais). Si ce service est arrêté, il n'y a pas moyen de pouvoir extraire les blocs d'un fichier donné. Pour répondre à cette problématique, un Namenode secondaire appelé Secondary Namenode a été mis en place dans l'architecture Hadoop. Son fonctionnement est relativement simple puisque le Namenode secondaire vérifie périodiquement l'état du Namenode principal et copie les métadonnées via les fichiers edits_xxx et fsimage_xxx. Si le Namenode principal est indisponible, le Namenode secondaire prend sa place.

3.3 Datanode

Précédemment, nous avons vu qu'un Datanode contient les blocs de données. Les Datanodes sont

sous les ordres du Namenode et sont surnommés les Workers. Ils sont donc sollicités par les Namenodes lors des opérations de lecture et d'écriture. En lecture, les Datenodes vont transmettre au client les blocs correspondant au fichier à transmettre. En écriture, les Datanodes vont retourner l'emplacement des blocs fraîchement créés. Les Datanodes sont également sollicités lors de l'initialisation du Namenode et aussi de manière périodique, afin de retourner la liste des blocs stockés.

4 MapReduce

MapReduce adresse deux choses. La première concerne le modèle de programmation MapReduce, étudié dans cette section. La seconde concerne le framework d'implémentation MapReduce, étudié dans le prochain article. Pour ce dernier, nous nous focaliserons sur les différentes API proposées par Apache Hadoop pour développer un programme MapReduce à partir du langage Java.

Le modèle de programmation fournit un cadre à un développeur afin d'écrire une fonction de Map et de Reduce. Tout l'intérêt de ce modèle de programmation est de simplifier la vie du développeur. Ainsi, ce développeur n'a pas à se soucier du travail de parallélisation et de distribution du travail, de récupération des données sur HDFS, de développements spécifiques à la couche réseaux pour la communication entre les nœuds, ou d'adapter son développement en fonction de l'évolution de la montée en charge (scalabilité horizontale, par exemple). Ainsi, le modèle de programmation permet au développeur de ne s'intéresser qu'à la partie algorithmique. Il transmet alors son programme MapReduce développé dans un langage de programmation au framework Hadoop pour l'exécution.

Autre chose avant de continuer, le terme de « job » MapReduce est couramment utilisé dans la littérature. Celui-ci concerne une unité de travail que le client souhaite réaliser. Cette unité est constituée de données d'entrée (contenues dans HDFS), d'un programme MapReduce (implémentation des fonctions map et reduce) et de paramètres d'exécution. Hadoop exécute ce job en le subdivisant en deux tâches : les tâches de Map et les tâches de Reduce.

Voyons maintenant le principe général de MapReduce, puis nous détaillerons son fonctionnement distribué dans Hadoop.

4.1 MapReduce : principe général

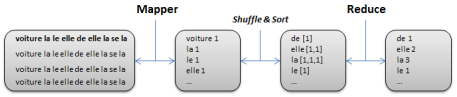
Les concepts de *map* et de *reduce* ne sont pas nouveaux puisqu'ils ont été empruntés aux langages fonctionnels, sauf que Google (lien 99) les a efficacement propulsés dans l'univers du calcul distribué et du grand volume de données. Ils sont utilisés pour implémenter des opérations de base sur les données comme le *tri*, le *filtrage*, la *projection*, l'*agrégation* ou le *regroupement*.

Pour expliquer les concepts de *map* et de *reduce*, partons de l'exemple du compteur de mots fréquemment utilisés, avec une légère variante. Tous les mots sont comptabilisés à l'exception du mot « se ». Ci-dessous, le fichier *exemple.txt* présente un jeu de données comportant une seule ligne.

```
1 voiture la le elle de elle la se la
      maison voiture
2 ...
```

On distingue clairement sur la première ligne que le mot *la* est répété trois fois et que le mot *voiture* est répété deux fois.

La figure ci-dessous énumère les différentes étapes qui seront présentées par la suite.



4.1.a Fonction map

La fonction map s'écrit de la manière suivante : map(clé1, valeur1) -> List(clé2, valeur2). À partir d'un couple clé/valeur, la fonction map retourne un ensemble de nouveaux couples clé/valeur. Cet ensemble peut être vide, d'une cardinalité un ou plusieurs.

Dans notre exemple, la *clé* d'entrée correspond au numéro de ligne dans le fichier et la *valeur* vaut *voiture la le elle de elle la se la maison voiture*.

Le résultat de la fonction *map* est donné ci-dessous.

```
1  (voiture,1)
2  (la,1)
3  (le,1)
4  (elle,1)
5  (de,1)
6  (elle,1)
7  (la,1)
8  (la,1)
9  (maison,1)
10 (voiture,1)
```

Nous remarquons que la fonction *map* retourne bien une liste de couples clé/valeur et que la clé utilisée à l'entrée de la fonction *map* n'est pas exploitée. Comme prévu, nous constatons que le mot « se » n'apparaît pas à la sortie de la fonction.

```
1 public class WordCountMapper extends
      Mapper<LongWritable, Text, Text,
      IntWritable> {
2     public void map(LongWritable key,
          Text value, Context context)
          throws IOException,
          InterruptedException {
```

```
3           StringTokenizer itr = new
                StringTokenizer(value.
                toString());
4           while (itr.hasMoreTokens()) {
5               String string = itr.
                    nextToken();
6               if (!string.equals("se")) {
7                   Text word = new Text();
8                   word.set(string);
9                   context.write(word, new
                        IntWritable(1));
10              }
11          }
12      }
13  }
```

Sans détailler l'API, puisque nous le ferons dans le prochain article, à la ligne 3 un StringTokenizer est utilisé pour découper la chaîne de caractères. Pour chaque mot obtenu, nous créons un couple dont la *clé* est le mot et la *valeur* vaut 1. Notez qu'à la ligne 6, nous effectuons un filtre pour exclure les instances du mot « se ».

Avant de présenter la fonction reduce, deux opérations intermédiaires doivent être exécutées pour préparer la valeur de son paramètre d'entrée. La première opération appelée shuffle permet de grouper les valeurs dont la clé est commune. La seconde opération appelée sort permet de trier par clé. À la différence des fonctions map et reduce, shuffle et sort sont des fonctions fournies par le framework Hadoop. Il n'y a donc pas à les implémenter.

Ainsi, après l'exécution des fonctions shuffle et sort le résultat de l'exemple est le suivant.

```
1  (de,[1])
2  (elle,[1,1]
3  (la,[1,1,1])
4  (le,[1])
5  (voiture,[1,1])
6  (maison,[1])
```

4.1.b Fonction reduce

La fonction reduce s'écrit de la manière suivante : reduce(clé2, List(valeur2)) -> List(valeur2). À partir des groupes de valeurs associées à une clé, la fonctions reduce retourne généralement une valeur ou aucune, bien qu'il soit possible de retourner plus d'une valeur.

Suite à l'appel de la fonction reduce, le résultat de l'exemple est le suivant.

```
1  (de,1)
2  (elle,2)
3  (la,3)
4  (le,1)
5  (voiture,2)
6  (maison,1)
```

Nous constatons que pour chaque clé, la fonction reduce effectue une somme de chaque élément de la liste.

```
1  public class WordCountReducer extends
       Reducer<Text, IntWritable, Text,
       IntWritable> {
2      public void reduce(Text key,
           Iterable<IntWritable> values,
           Context context) throws
           IOException,
           InterruptedException {
3          int sum = 0;
4          for (IntWritable current :
               values) {
5              sum += current.get();
6          }
7          context.write(key, new
               IntWritable(sum));
8      }
9  }
```

Rien de bien méchant dans l'implémentation de la fonction reduce. Le parcours des éléments de la liste de valeurs se fait à la ligne 4 depuis l'itérateur values. Pour chaque élément de la liste, une somme est effectuée à la ligne 5.

4.2 MapReduce dans Hadoop

Intéressons-nous maintenant à voir comment le modèle de programmation MapReduce est exploité dans le framework Hadoop. Je tiens avant tout à préciser que je ne détaillerai pas le fonctionnement du nouveau système YARN pour la gestion de jobs MapReduce, car je n'ai pas encore assez de recul. Je peux juste faire remarquer que les composants *JobTracker* et *TaskTrackers* ont été remplacés par les composants *ResourceManager*, *NodeManager* et *ApplicationMaster*.

4.2.a Étape du découpage des données (split)

La première étape concerne le découpage (split) des données. Cette étape est à la charge du framework qui se base sur le format des données. La taille de chaque découpage est fixe et généralement identique à la taille d'un bloc HDFS (64 Mo par défaut).

En s'appuyant sur l'exemple du compteur de mots fréquemment utilisés, le découpage consistera à réaliser des morceaux de 64 Mo qui contiendront chacun une liste de clé/valeur où la clé sera l'offset d'une ligne du fichier et la valeur le contenu de cet offset (*voiture la le elle de elle la se la maison voiture*).

4.2.b Étape Map

Hadoop crée pour chaque découpage de données une tâche de Map qui exécutera la fonction map développée en conséquence. Chaque découpage de données n'est traité que par une seule tâche de Map.

Précisons que ce n'est pas la donnée qui est transportée vers le programme, mais l'inverse. Cela permet de pouvoir profiter d'une optimisation appelée **optimisation de proximité de la donnée** (Data Locality Optimization en anglais). Hadoop va donc tenter de trouver le nœud le plus proche contenant la donnée pour y transférer la fonction Map.

Une fois la tâche Map terminée, le résultat de la fonction n'est pas stocké sur le système HDFS, mais sur le système de fichier local du nœud. En effet, il s'agit d'un résultat intermédiaire et qui n'a pas besoin d'être stocké de manière sécurisée via la réplication des blocs HDFS. Dans le cas d'une anomalie sur l'exécution de la tâche (nœud qui devient inaccessible par exemple), Hadoop est informé et pourra demander sa rééxécution.

4.2.c Étape Reduce

Hadoop ne lance les tâches de Reduce qu'une fois que toutes les tâches de Map sont terminées. Notez que le nombre de tâches de Reduce n'est pas fonction de la taille des données en entrée mais est spécifié en paramètre de configuration d'exécution du job. C'est donc un paramètre qui peut être modifié. Par défaut, le nombre de tâches de Reduce est de un.

L'entrée de la fonction reduce correspond à la sortie de l'ensemble des fonctions map. Comme la tâche de Reduce ne profite pas d'optimisation de proximité comme cela est fait pour la tâche de Map, les données de sortie de la fonction map sont transférées via le réseau vers le nœud où la tâche de Reduce est réalisée.

Chaque tâche de Reduce produit un fichier de sortie qui sera stocké, cette fois, dans le système de fichiers HDFS. Le format du nom du fichier produit est de la forme suivante : *part-r-XXXX* où *XXXX* est le numéro de la tâche de Reduce commençant par 0. Dans notre exemple, il y aura donc un seul fichier, car une seule tâche de Reduce, qui portera le nom part-r-0000.

```
1  de     1
2  elle     2
3  la     3
4  le     1
5  voiture     2
6  maison     1
```

4.2.d Étape Combiner (facultative)

Précédemment, nous avons vu que les données de sortie de la fonction map étaient transférées par le réseau vers le nœud où s'effectuera la tâche de Reduce. Bien entendu, cela peut avoir un impact sur les performances car les données à transmettre peuvent être volumineuses. Pour répondre à ce problème, une autre optimisation consiste à utiliser une fonction dite *Combiner* en sortie directe de la fonction map. Le Combiner a le même comportement que la tâche Reduce et il s'appuie sur la même API.

Quand un Combiner est utilisé, la sortie de la fonction map n'est pas écrite sur le système de fichiers local. Un traitement en mémoire est réalisé afin de regrouper les valeurs par clé (identique à shuffle et sort).

Considérons la sortie d'une première tâche *Map* (identique à notre précédent exemple).

```
1  (voiture,1)
2  (la,1)
3  (le,1)
4  (elle,1)
5  (de,1)
6  (elle,1)
7  (la,1)
8  (la,1)
9  (maison,1)
10 (voiture,1)
```

Considérons maintenant la sortie d'une seconde tâche *Map*.

```
1  (voiture,1)
2  (voiture,1)
3  (voiture,1)
```

Après application de la fonction Combiner sur la sortie de la première tâche *Map*, le résultat attendu est le suivant.

```
1  (de,1)
2  (elle,2)
3  (la,3)
4  (le,1)
5  (voiture,2)
6  (maison,1)
```

De même, après application de la fonction Combiner sur la sortie de la seconde tâche *Map*, le résultat attendu est le suivant.

```
1  (voiture,3)
```

Par conséquent, l'entrée de la fonction reduce sera celui-ci.

```
1  (de,[1])
2  (elle,[2])
3  (la,[3])
4  (le,[1])
5  (voiture,[2,3])
6  (maison,[1])
```

5 Écosystème Hadoop : une foire aux sous-projets

Comme expliqué précédemment, Hadoop est un système distribué orienté *batch*, taillé pour le traitement de jeux de données volumineux. Les utilisateurs d'Hadoop se retrouvent alors à manipuler le système de fichiers HDFS ou à développer des programmes MapReduce bas niveau en partant souvent de rien. Des sous-projets à Hadoop sont nés de ce constat et offrent des mécanismes et fonctionnalités qui simplifient la manipulation et le traitement des jeux de données volumineux. Nous en présenterons brièvement quelques-uns dans cette section. Une liste complète peut être trouvée ici : Bigdata Ecosystem (lien 100)

5.1 HBase

HBase (lien 101) permet l'intégration à Hadoop d'un système de stockage par clé/valeur appelé couramment stockage binaire ou key/value store en anglais.

Ce sous-projet à Hadoop est également inspiré par le projet BigTable (lien 102) de Google.

5.2 Hive

Hive (lien 103) crée une base de données relationnelle dans le système de fichiers HDFS. Le projet permet aux développeurs d'écrire des requêtes, dans un langage proche de SQL appelé HiveQL, qui sont ensuite traduites comme des programmes MapReduce sur le cluster. L'avantage est de pouvoir fournir un langage que les développeurs connaissent pour l'écriture des programmes MapReduce.

5.3 Pig

Le projet Pig (lien 104) se positionne comme Hive dans le sens où il fournit aux développeurs un langage de haut niveau (un DSL) dédié à l'analyse de gros volumes de données. Il s'adresse alors aux développeurs habitués à faire des scripts via Bash ou Python par exemple. Par ailleurs, Pig est extensible dans le sens où, si une fonction n'est pas disponible, il est possible de l'enrichir via des développements spécifiques dans un langage bas niveau (Java, Python...).

Dans le même ordre d'idées que le projet Pig, il y a Scalding (lien 105) qui puise la puissance du langage Scala pour développer ses programmes MapReduce.

5.4 Sqoop

Sqoop (lien 106) est un projet qui aide à dialoguer avec des systèmes de gestion de base de données relationnelle vers Hadoop. Le projet permet d'importer et d'exporter des données de ou vers une base de données.

5.5 Mahout

Mahout (lien 107) fournit des implémentations d'algorithmes pour faire de l'informatique décisionnelle. Il fournit par exemple des algorithmes pour faire du partitionnement de données ou de la classification automatique dans un environnement MapReduce.

6 Jeux de données

Le problème pour tester Apache Hadoop est de disposer de jeux de données réalistes. Je vous propose, dans cette section, un recueil de sites qui proposent différents types de données.

- The National Bureau of Economic Research (lien 108) : regroupe plus de trois millions de brevets américains déclarés pendant la période de janvier 1963 à décembre 1999.
- Données publiques depuis Amazon Web Services (lien 109) : regroupe un ensemble de données rendues publiques comme **NASA NEX**, une collection d'ensembles de données portant sur les sciences de la Terre entretenue par la NASA, **Common Crawl Corpus**, un corpus de données d'indexation Web composé de plus de cinq milliards de pages Web, ou encore des **données du recensement des États-Unis** pour la période 1980, 1990 et 2000.
- ClueWeb09 (lien 110) : contient plus d'un milliard de pages Web dédiées à la recherche sur des technologies du langage humain (lien 111).
- IMDB : un sous-ensemble de la base de données liée au monde du cinéma.
- National Climatic Data Center (lien 112) : données concernant le climat proposées par la National Oceanic and Atmospheric Administration (lien 113), une institution gouvernementale des États-Unis.
- Grouplens (lien 114) : groupe de recherche, dans le domaine de systèmes de recommandation et d'interaction homme-machine, qui a collecté de grands volumes de données comme par exemple sur des classements de films auprès d'utilisateurs.
- DBPedia (lien 115) : version structurée de Wikipédia

7 Conclusion, à suivre dans le prochain article

Cet article s'est intéressé à présenter les généralités d'Hadoop, HDFS et le modèle de programmation MapReduce. La présentation des concepts était obligatoire afin de mieux cerner les articles plus techniques qui vont suivre.

Dans le prochain tutoriel de cette série, nous montrerons comment installer et configurer un cluster simple nœud avec Cloudera CDH. L'intérêt d'un cluster simple nœud a pour vocation d'être pédagogique (comprendre les fichiers de configuration, les outils de base...) et de disposer d'un environnement de développement pour tester ses développements de programmes MapReduce.

8 Références

8.1 Cours, Articles

- Explication sur Map/Reducce : lien 116.
- Des jeux de données pour stresser son cluster : lien 117.
- Le site du Hadoop User Group français : lien 118.
- Un billet qui présente les prérequis avant de foncer vers Hadoop : lien 119.
- Hadoop sur Wikipédia : lien 120.
- Supports de cours sur Hadoop : lien 121.
- Des billets sur Hadoop : lien 122.
- Une présentation sur HDFS : lien 123.
- Une série de billets sur Hadoop : lien 124.
- Un livre Open Source sur MapReduce : lien 125.
- Un billet qui présente les nouveautés Hadoop 2 : lien 126.

8.2 Livres

- Hadoop Illuminated : lien 127.
- Hadoop In Pratice : lien 128.
- Hadoop Operations : lien 129.
- Hadoop The Definitive Guide : lien 130.
- Hadoop In Action : lien 131.

*Retrouvez l'article de **Mickael Baron** en ligne : lien 132*

Liste des liens

Page 2
lien 1 : ... http://java.developpez.com/
lien 2 : ... http://javascript.developpez.com/cours/
lien 3 : ... http://python.developpez.com/cours/

Page 4
lien 4 : ... http://vviale.developpez.com/tutoriels/openoffice-libreoffice/installation/#LIV-C

Page 10
lien 5 : ... http://openoffice-libreoffice.developpez.com/livres/index/?page=Suite-bureautique#L2212132476
lien 6 : ... http://vviale.developpez.com/tutoriels/openoffice-libreoffice/apprendre-obasic-tableur-calc/

Page 15
lien 7 : ... http://vviale.developpez.com/tutoriels/openoffice-libreoffice/diagramme-gantt/

Page 16
lien 8 : ... https://www.salesforce.com/fr/company/
lien 9 : ... http://fr.wikipedia.org/wiki/Logiciel_en_tant_que_service
lien 10 : ... http://fr.wikipedia.org/wiki/Plate-forme_en_tant_que_service
lien 11 : ... http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_triggers.htm
lien 12 : ... http://fr.wikipedia.org/wiki/Web_Services_Description_Language
lien 13 : ... http://developer.force.com/
lien 14 : ... http://www.salesforce.com/crm/editions-pricing.jsp

Page 21
lien 15 : ... http://aurelien-laval.fr/tutorial/salesforce-appeler-un-webservice-externe-depuis-salesforce-19

Page 23
lien 16 : ... http://aurelien-laval.developpez.com/tutoriels/salesforce/tester-son-code-apex/fichiers/sources_tutoriel_salesforce_4.zip
lien 17 : ... https://github.com/AurelienLaval/tester-son-code-apex-dans-salesforce
lien 18 : ... https://bitbucket.org/AurelienLaval/tester-son-code-apex-dans-salesforce
lien 19 : ... http://wiki.developerforce.com/page/An_Introduction_to_Apex_Code_Test_Methods
lien 20 : ... http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_testing_example.htm
lien 21 : ... http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_callouts_wsdl2apex_testing.htm
lien 22 : ... http://saramorgan.net/2013/08/28/more-about-web-services-and-unit-testing/
lien 23 : ... http://help.salesforce.com/HTViewHelpDoc?id=code_dev_console_tests_coverage.htm&language=en_US
lien 24 : ... http://www.salesforce.com/us/developer/docs/apexcode/Content/apex_testing_best_practices.htm
lien 25 : ... http://aurelien-laval.developpez.com/tutoriels/salesforce/tester-son-code-apex/

Page 24
lien 26 : ... http://opengl.developpez.com/docs/man/man/glmapbuffer
lien 27 : ... http://opengl.developpez.com/docs/man/man/glBufferSubData

Page 25
lien 28 : ... http://opengl.developpez.com/docs/man/man/gltexstorage2d

Page 26
lien 29 : ... http://www.developpez.net/forums/d1439426/applications/developpement-2d-3d-jeux/verite-qualite-pilotes-graphiques-opengl/

Page 27
lien 30 : ... http://nicoboo.developpez.com/articles/xna/presentation/

lien 31 : ... http://jeux.developpez.com/actu/51518/Microsoft-ne-travaille-plus-sur-XNA-la-technologie-sera-retiree-du-programme-Most-Valuable-Professional-le-1er-avril/
lien 32 : ... http://blog.developpez.com/jeux/p12527/developpement-2d-3d-jeux/kodu-game-lab-sous-windows-8-grace-a-monogame

Page 28
lien 33 : ... http://www.developpez.net/forums/d1436210/applications/developpement-2d-3d-jeux/xna-mort-on-s-interesser-technologie-microsoft-faire-jeux/

Page 29
lien 34 : ... http://jeux.developpez.com/evenements/presentations-projets/4/#LII-J
lien 35 : ... https://www.youtube.com/v/ipJUBIhLVM4&autoplay=1
lien 36 : ... https://www.youtube.com/v/ZqlveWIhCFI&autoplay=1
lien 37 : ... https://www.youtube.com/v/DOUSL6Za3SU&autoplay=1
lien 38 : ... http://www.qb64.net/
lien 39 : ... http://www.qb64.net/

Page 30
lien 40 : ... http://www.qb64.net/

Page 34
lien 41 : ... http://alexandre-laurent.developpez.com/tutoriels/QB64/installation-introduction/

Page 35
lien 42 : ... http://www.docbook.org/
lien 43 : ... https://www.debian.org/
lien 44 : ... http://www.inetdoc.net/pdf/ethernet.pdf

Page 44
lien 45 : ... http://inetdoc.developpez.com/tutoriels/technologie-ethernet/

Page 47
lien 46 : ... http://docs.oracle.com/javase/8/javase-clienttechnologies.htm
lien 47 : ... http://www.oracle.com/technetwork/java/javase/downloads/index.html
lien 48 : ... http://www.developpez.net/forums/d1430843/java/interfaces-graphiques-java/javafx/java-8-apporte-nouveautes-cote-java-fx-venez-decouvrir-ameliorations-javafx-8-a/

Page 48
lien 49 : ... https://www.youtube.com/watch?v=ByDPifJMSvQ
lien 50 : ... http://www.developpez.net/forums/d1435767/java/general-java/langage/scala/scala-passe-version-2-11-a/

Page 49
lien 51 : ... http://disqus.com/gavinking/
lien 52 : ... http://lmauzaize.developpez.com/tutoriels/ceylon/presentation/

Page 50
lien 53 : ... http://modules.ceylon-lang.org/repo/1/ceylon/collection/1.0.0/module-doc/index.html

Page 52
lien 54 : ... https://docs.jboss.org/author/display/MODULES/Introduction
lien 55 : ... http://modules.ceylon-lang.org/
lien 56 : ... http://ceylon-lang.org/documentation/current/reference/tool/ceylon/subcommands/ceylon-compile.html
lien 57 : ... http://daringfireball.net/projects/markdown/syntax

Page 53
lien 58 : ... http://modules.ceylon-lang.org/repo/1/ceylon/language/1.0.0/module-doc/true.object.html
lien 59 : ... http://modules.ceylon-lang.org/repo/1/ceylon/language/1.0.0/module-doc/false.object.html
lien 60 : ... http://modules.ceylon-lang.org/repo/1/ceylon/language/1.0.0/module-doc/Boolean.type.html
lien 61 : ... http://modules.ceylon-lang.org/repo/1/ceylon/language/1.0.0/module-doc/Integer.type.html

lien 62 : ... http://modules.ceylon-lang.org/repo/1/ceylon/language/1.0.0/module-doc/Float.type.html

Page 54

lien 63 : ... http://www.unicode.org/Public/UNIDATA/NamesList.txt

Page 58

lien 64 : ... http://fr.wikipedia.org/wiki/Fermeture_(informatique)

Page 60

lien 65 : ... http://lmauzaize.developpez.com/tutoriels/ceylon/bases/

Page 67

lien 66 : ... https://github.com/leleuj/spring-security-pac4j/blob/master/src/main/java/org/pac4j/springframework/security/web/ClientAuthenticationFilter.java

lien 67 : ... https://github.com/leleuj/spring-security-pac4j/blob/master/src/main/java/org/pac4j/springframework/security/authentication/ClientAuthenticationProvider.java

lien 68 : ... http://thierry-leriche-dessirier.developpez.com/tutoriels/java/pac4j-5-min/

Page 68

lien 69 : ... http://mbaron.developpez.com/tutoriels/nosql/hadoop/introduction-hdfs-map-reduce

lien 70 : ... http://mbaron.developpez.com/tutoriels/nosql/hadoop/installation-configuration-cluster-singlenode-avec-cloudera-cdh5/

lien 71 : ... http://nutch.apache.org/

lien 72 : ... http://lucene.apache.org/

lien 73 : ... http://research.google.com/archive/gfs.html

lien 74 : ... http://research.google.com/archive/mapreduce.html

lien 75 : ... http://hadoop.apache.org/

Page 69

lien 76 : ... http://www.chrisstucchio.com/blog/2013/hadoop_hatred.html

lien 77 : ... http://fr.yahoo.com/?p=us

lien 78 : ... http://wiki.apache.org/hadoop/FAQ#How_well_does_Hadoop_scale.3F

lien 79 : ... http://wiki.apache.org/hadoop/PoweredBy

lien 80 : ... http://hadoopilluminated.com/hadoop_book/Hadoop_Use_Cases.html

lien 81 : ... http://hugfrance.fr/

lien 82 : ... http://www.cloudera.com/content/dam/cloudera/Resources/PDF/Cloudera_Case_Study_Healthcare.pdf

lien 83 : ... http://www.nextbio.com/

lien 84 : ... http://hadoop.intel.com/pdfs/IntelNextBioCaseStudy.pdf

lien 85 : ... http://www.nokia.com/

lien 86 : ... http://www.cloudera.com/content/dam/cloudera/Resources/PDF/Cloudera_Nokia_Case_Study_Hadoop.pdf

lien 87 : ... http://france.edf.com/

lien 88 : ... http://hugfrance.fr/meetup-13-fevrier-2014/

lien 89 : ... http://perso.isep.fr/rchiky/bigdata/MLP.pdf

lien 90 : ... http://www.usxpress.com/

lien 91 : ... http://hortonworks.com/wp-content/uploads/downloads/2013/06/Hortonworks.BusinessValueofHadoo v1.0.pdf

lien 92 : ... http://java.dzone.com/articles/can-mapreduce-solve-planning

lien 93 : ... http://fr.wikipedia.org/wiki/Problème_du_voyageur_de_commerce

lien 94 : ... http://www.etsy.com/

lien 95 : ... http://www.slideshare.net/mwalkerinfo/funnel-analysis-in-hadoop-at-etsy

lien 96 : ... http://www.skyboximaging.com/

lien 97 : ... http://blog.cloudera.com/blog/2012/10/sneak-peek-into-skybox-imagings-cloudera-powered-satellite-system/

Page 70

lien 98 : ... http://blog.cloudera.com/blog/2009/02/the-small-files-problem/

Page 71

lien 99 : ... http://research.google.com/archive/mapreduce.html

Page 73

lien 100 : ... http://hadoopilluminated.com/hadoop_illuminated/Bigdata_Ecosystem.html

Page 74

lien 101 : ... http://hbase.apache.org/

lien 102 : ... http://static.googleusercontent.com/media/research.google.com/fr//archive/bigtable-osdi06.pdf

lien 103 : ... http://hive.apache.org/

lien 104 : ... http://pig.apache.org/

lien 105 : ... https://github.com/twitter/scalding

lien 106 : ... http://sqoop.apache.org/

lien 107 : ... http://mahout.apache.org/

Page 75

lien 108 : ... http://www.nber.org/patents/

lien 109 : ... http://aws.amazon.com/fr/publicdatasets/

lien 110 : ... http://www.imdb.com/interfaces

lien 111 : ... http://lemurproject.org/clueweb09/

lien 112 : ... http://www.ncdc.noaa.gov/data-access

lien 113 : ... http://www.noaa.gov/

lien 114 : ... http://grouplens.org/datasets

lien 115 : ... http://wiki.dbpedia.org/

lien 116 : ... http://wiki.apache.org/hadoop/HadoopMapReduce

lien 117 : ... http://www.hadooplessons.info/2013/06/data-sets-for-practicing-hadoop.html

lien 118 : ... http://hugfrance.fr/

lien 119 : ... http://blog.octo.com/votre-premier-projet-hadoop/

lien 120 : ... http://en.wikipedia.org/wiki/Apache_Hadoop

lien 121 : ... http://courses.coreservlets.com/Course-Materials/pdf/hadoop/

lien 122 : ... http://blog.inovia-conseil.fr/?cat=27

lien 123 : ... http://fr.slideshare.net/hugfrance/introduction-hdfs

lien 124 : ... http://www.bigdataplanet.info/search/label/Hadoop-Tutorial

lien 125 : ... http://lintool.github.io/MapReduceAlgorithms

lien 126 : ... http://strata.oreilly.com/2014/01/an-introduction-to-hadoop-2-0-understanding-the-new-data-operating-system.html

lien 127 : ... http://hadoopilluminated.com/

lien 128 : ... http://www.manning.com/holmes/

lien 129 : ... http://shop.oreilly.com/product/0636920025085.do

lien 130 : ... http://shop.oreilly.com/product/0636920021773.do

lien 131 : ... http://www.manning.com/lam/

lien 132 : ... http://mbaron.developpez.com/tutoriels/nosql/hadoop/introduction-hdfs-map-reduce/