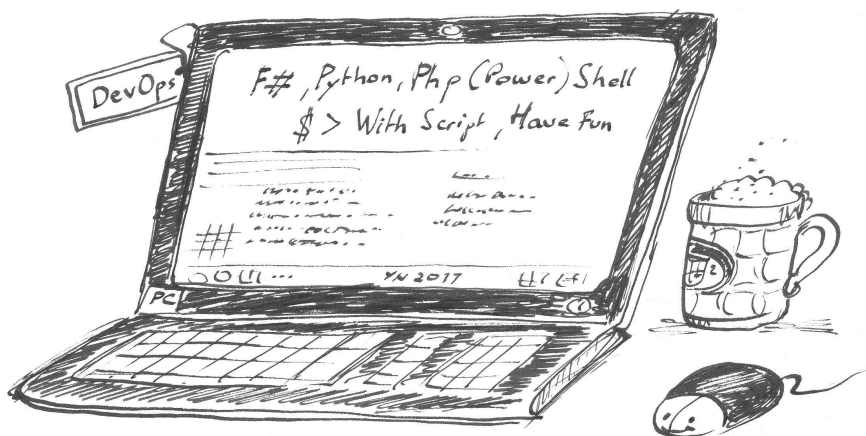


Pierre
Contri

Analysez (vos besoins), Etudiez, Scriptez



L'art d'utiliser du scripting pour un projet

Janvier 2017

ANALYSEZ (VOS BESOINS),
ETUDIEZ, SCRIPTEZ

Pierre Contri

Analysez (vos besoins),
Etudiez, Scriptez

© Pierre Contri, 2017

Toute reproduction ou publication, même partielle, de cet ouvrage est interdite sans l'autorisation préalable de l'auteur.

A ma famille

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

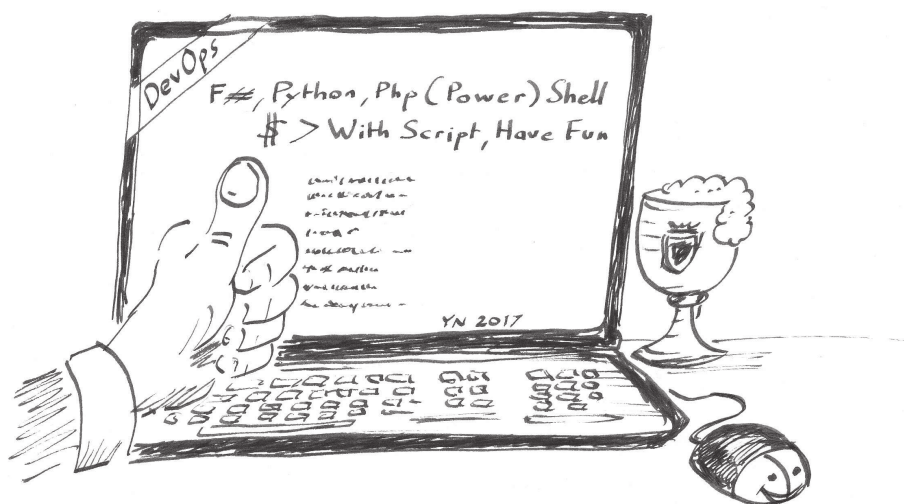
Table des matières

| | |
|--|----|
| Introduction..... | 11 |
| Concepts Généraux..... | 12 |
| Orientation intellectuelle..... | 13 |
| Outils rudimentaires et indispensables..... | 14 |
| Travailler avec les octets, chaînes de caractères et les tableaux..... | 15 |
| Les octets (bytes)..... | 16 |
| Les chaînes de caractères (strings)..... | 18 |
| Les Tableaux (arrays)..... | 19 |
| Différents langages de scripting..... | 29 |
| PHP..... | 31 |
| Pourquoi utiliser ce langage..... | 32 |
| Afficher simplement 'Hello World'..... | 33 |
| Comment utiliser les scripts par la console..... | 34 |
| Serveur Web en script PHP..... | 34 |
| Aller plus loin avec le scripting en mode console..... | 41 |
| Interaction entre le Scripting et une interface graphique..... | 50 |
| Gestion de la plate-forme hôte..... | 52 |
| Exemple avec le programme « Picture Resizing »..... | 53 |
| Python..... | 65 |
| Pourquoi utiliser ce langage..... | 66 |
| Afficher simplement 'Hello World'..... | 67 |
| Comment utiliser les scripts par la console..... | 68 |
| Aller plus loin avec le Scripting en mode console..... | 69 |
| Interaction entre le Scripting et une interface graphique..... | 85 |
| Utiliser GTK+ et Python..... | 87 |

Analysez (vos besoins), Etudiez, Scriptez
L'art d'utiliser du scripting pour un projet

| | |
|--|-----|
| Exemple de programme: Car Cost Simulator..... | 87 |
| Javascript..... | 137 |
| Pourquoi utiliser ce langage..... | 138 |
| Afficher simplement 'Hello World'..... | 139 |
| Comment utiliser les scripts par la console..... | 140 |
| Aller plus loin avec le Scripting en mode console..... | 141 |
| Interaction entre le Scripting et une interface graphique..... | 146 |
| Le Pong..... | 147 |
| Le Pong2..... | 150 |
| PowerShell..... | 167 |
| Pourquoi utiliser ce langage..... | 168 |
| Afficher simplement 'Hello World'..... | 169 |
| Comment utiliser les scripts par la console..... | 170 |
| Aller plus loin avec le Scripting en mode console..... | 171 |
| Travailler avec Azure et la partie System..... | 173 |
| Interaction entre le Scripting et une interface graphique..... | 183 |
| Perl..... | 189 |
| Pourquoi utiliser ce langage..... | 190 |
| Afficher simplement 'Hello World'..... | 191 |
| Comment utiliser les scripts par la console..... | 192 |
| Aller plus loin avec le Scripting en mode console..... | 193 |
| Interaction entre le Scripting et une interface graphique..... | 195 |
| F# code et script..... | 197 |
| Pourquoi utiliser ce langage..... | 198 |
| Afficher simplement 'Hello World'..... | 199 |
| Comment utiliser les scripts par la console..... | 200 |
| Aller plus loin avec le Scripting en mode console..... | 202 |
| Interaction entre le Scripting et une interface graphique..... | 206 |
| Lien entre tous ces langages de script..... | 219 |
| Créer une DLL et l'utiliser dans notre script..... | 220 |
| Utiliser une DLL et travailler avec en PowerShell..... | 221 |
| Utiliser une DLL et travailler avec en PHP..... | 223 |
| Use Scripting..... | 225 |
| Compiler les scripts? Pourquoi faire?..... | 226 |
| Compiler du Python en tant que ".pyc"..... | 227 |
| Compiler des scripts? OK. Utilisons d'autres langages..... | 228 |
| Convertir Python en C/C++..... | 228 |
| Utiliser DotNet C#, F# ou Java..... | 228 |

| | |
|---|-----|
| Compatibilité..... | 229 |
| Scripts: aller vers la programmation fonctionnelle..... | 230 |
| Conclusion..... | 230 |
| Bye World !..... | 230 |
| Remerciements..... | 231 |
| Bibliographie..... | 232 |
| Site Web de l'auteur..... | 232 |
| Contact..... | 232 |



Introduction

J'ai écrit ce livre dans un esprit de recherche et d'évolution intellectuelle. J'aime par-dessus tout expérimenter des solutions techniques et analyser ce qui peut se faire de mieux pour un cahier des charges précis tant en solution de Concept que de Production. Un seul objectif, la progression intellectuelle.

L'écriture pourrait sembler décousue à certains moments, cela est dû au fait que je fractionne mon travail d'écriture en morceaux de 20 minutes à 1 heure maximum, résultant de mon temps libre; excusez-moi par avance si le style n'est pas fluide.

Contenu de ce chapitre

| | |
|--|----|
| Introduction..... | 11 |
| Concepts Généraux..... | 12 |
| Oritentation intellectuelle..... | 13 |
| Outils rudimentaires et indispensables..... | 14 |
| Travailler avec les octets, chaînes de caractères et les tableaux..... | 15 |
| Les octets (bytes)..... | 16 |
| Les chaînes de caractères (strings)..... | 18 |
| Les Tableaux (arrays)..... | 19 |

Concepts Généraux

PHP, Python sont vraiment des langages très intéressants à apprendre et utiliser. On peut s'en servir pour des applications Web, clients légers, clients lourds ou même des solutions complètes.

Avec cette philosophie (ouverte et non restreinte basée sur des "à priori" ou des "c'est une technologie pour le web"), nous pouvons adapter n'importe quelle demande avec l'appui de n'importe quel langage.

Le développeur a tous les pouvoirs sur le développement qu'on lui demande. Le problème est qu'il développe ce qu'il veut, comme il l'entend ; le logiciel alors développé peut être un logiciel architecturé ou une grosse bouse. Cela dépend de l'ouverture d'esprit de ce dernier et de son analyse au préalable.

Au final, même si l'analyse était bonne avant de débiter et que le développeur code de manière ponctuelle (sans pensée de pérennisation), le produit délivré peut être compromis.

Ce qui est drôle, c'est que le développeur n'utilise pas son outil. Par moment, il ne peut même pas se rendre compte des absurdités qu'il produit. C'est comme un maçon qui construira sa maison avec une qualité intéressante et celle des autres avec une qualité médiocre. Tout comme un ouvrier à la chaîne, qui va assembler des pièces sans s'y intéresser. Si on fait assembler à un ouvrier son propre produit, je suis persuadé que le produit sera de meilleur facture.

Pour ma part, j'utilise mes "produits", ce qui fait que j'essaie au maximum que possible de fournir un logiciel non bogué, quitte donner des délais longs, mais toujours respectés.

Orientation intellectuelle

La problématique de cet écrit est que je voudrais présenter à la fois un livre technique et à la fois sur une gestion de projet.

La gestion de projet ne serait pas en Mode La Rache ®, mais avec un minimum de spécifications, tout en présentant plusieurs technologies.

En fait, se mettre dans la peau d'un Chef de Projet technique ou un architecte qui doit commencer un nouveau projet et qui doit trouver le bon rapport qualité / prix / analyse avec le langage le plus approprié, même si ce n'est pas le sien.

En bref, ce qui m'est arrivé avec Python en 2011. Je ne connaissais pas ce langage, je voulais faire un logiciel en .Net. Mon analyse après deux jours d'études me disait que ce n'était pas du tout le bon langage; tourné ensuite vers Java, l'analyse était trop compliquée, trop de temps, pas possible de développer par tranches de 20 minutes tous les 2 jours; et c'est là que j'ai découvert Python couplé à GTK+.

Objectivement, c'était le bon langage pour ce projet, même si j'ai dû apprendre ce langage ; et donc, par ce fait, dû investir du temps au démarrage de ce projet. Cela a payé par la suite, au vu des différentes caractéristiques ajoutées.

Outils rudimentaires et indispensables

Mon parcours professionnel m'a montré qu'il fallait se débrouiller avec très peu d'outils, c'est pour cette raison que je ne présenterai que peu d'outils et tous très simples et intégrés au système hôte.

Nommé "the A-Team" chez un client, j'ai intégré l'équipe de maintenance niveau 3 de ce dernier. Notre équipe était réellement le dernier recours, quand plus rien n'allait. Pour résoudre la majorité des incidents de production, mes collègues et moi-même nous connectons directement sur la production. Par exemple, une erreur dans une base de données, nous créons la requête SQL sur Notepad et nous exécutons cette dernière sur le serveur de production en ligne de commande. Autre exemple: une erreur d'utilisation d'un site SharePoint nous a contraints à créer un script PowerShell qui parcourait chaque document de la « site collection » pour l'extraire et le récupérer dans un dossier externe à SharePoint. Tout cela avec seulement les outils de Windows 2008R2, donc l'ISE.

Habitué à travailler toujours en flux tendus et sans réel temps pour créer un programme compilé avec tout ce qui va avec, je me suis spécialisé dans les scripts très rapides, tout en me fabriquant une boîte à outils de modules de scripts capables de me sortir du prochain pétrin que mon commercial allait me donner.

Travailler avec les octets, chaînes de caractères et les tableaux

Travailler avec des octets, chaînes de caractères et tableaux est la partie qui m'intéresse le plus en informatique avec la communication entre appareils.

Nous utilisons l'informatique pour traiter de l'information. Le plus petit dénominateur commun entre le traitement de l'information d'un robot d'assemblage de véhicule et d'une base de données contenant des informations bancaires est le tableau, voire l'octet. Être à l'aise avec le jeu de tableaux, de chaînes de caractères, permet de comprendre et de manipuler des objets en prenant de la hauteur et en imaginant toujours une généricité de ce que l'on souhaite.

La suite logique après l'utilisation des tableaux est l'orientation vers les tableaux complexes dont certains sont des dictionnaires.

Un dictionnaire peut être vu comme un tableau d'index nommés, auquel on y donne une valeur (simple ou complexe). Le dictionnaire permet de représenter une structure de données ou même un objet (exemple avec Json).

Les octets (bytes)

L'octet est la base de tout. Malheureusement, il est complètement inhibé par nos programmations "haut niveau" actuelles.

L'octet (contenant 8bits) est dérivé pour absolument tout. Il représente la plus petite information au niveau informatique "haut niveau".

Nous pouvons utiliser l'octet pour représenter un caractère, un nombre, un tableau de 8 états logiques, une adresse, Par Exemple:

```
myByte <- 079
```

L'octet "myByte" a la valeur 79.

Pour la machine, ce n'est qu'une valeur qui sera traitée numériquement. Cela veut dire que si je lui rajoute 1, la valeur passe à 80. Vous allez trouver cela logique et simple, mais si l'on va plus loin, on peut se dire que l'ajout de ce "1", peut correspondre à un changement de lettre, ou encore à un basculement de minimum 2 états logiques ! Sur un robot, il peut y avoir une grande incidence.

La valeur 79 de cet octet, converti en chaîne de caractère (encodage ASCII) correspond à "O".

En ajoutant "1", on passe donc à la valeur "P".

La même valeur 79 peut aussi être représentée en binaire, tel que :01001111.

Cette suite peut aussi être interprétée comme un tableau d'états logiques. En ajoutant "1", on passe à 01010000. Si cet octet était connecté à la gestion de 8 lampes dans une pièce, nous aurions pu allumer 1 lampe et en éteindre 4 ! Il est donc important et crucial de comprendre que le jeu avec un octet peu s'avérer dangereux si on ne le comprend pas. Pour rappel, chaque 0 ou 1 de l'octet peu être lié à une lampe dans notre exemple.

Notons au passage que, dans nos programmations évoluées, le type Boolean apparaît. Ce type est, la plupart du temps, codé sur deux octets (16 bits).

Ce type représente l'état logique vrai ou faux, soit 1 ou 0 côté machine.

Pour schématiser 00000000 00000001 représente vrai et 00000000 00000000 représente faux.

Avec une telle quantité d'informations (côté machine) pour traiter si peu d'informations (côté humain), on peut extrapoler cela à un logiciel, et comprendre pourquoi, rien que pour afficher une fenêtre graphique, nous avons besoin de 50 Mo de mémoire. Cela est un autre débat mais met bien en évidence la différence entre de l'informatique embarquée (où chaque bit compte) et l'informatique de bureau (où seul le temps de l'homme compte).

Les chaînes de caractères (strings)

Les chaînes de caractères sont le déguisement d'un tableau de caractères.

Certains langages ne fonctionnent qu'avec ce type de base; ces langages sont, en général des langages de Scripting.

La chaîne de caractères peut servir pour beaucoup de choses, par exemple, stocker une image en mémoire vive et l'envoyer par la liaison série RS232.

L'image (tableau d'octets) est encodée en Base 64 pour représenter une chaîne de caractères qui sera transmise caractère par caractère en liaison série.

Exemple pour un pixel rouge transcodé en une chaîne de caractères:

```
myPixel <- [ 255, 0, 0]
myArrayBase64 <- ConvertToBase64(myPixel)
```

```
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. Tous droits réservés.

PS C:\Users\Pierre> $myPixel = @(255, 0, 0)
PS C:\Users\Pierre> $myPixel
255
0
0
PS C:\Users\Pierre> $myArrayBase64 = [System.Convert]::ToBase64String($myPixel)
PS C:\Users\Pierre> $myArrayBase64
/wAA
PS C:\Users\Pierre> $myArrayBase64.Length
4
PS C:\Users\Pierre> _
```

```
pierre@debian-work2: ~
Fichier Édition Onglets Aide
pierre@debian-work2:~$ python
Python 2.7.9 (default, Mar 1 2015, 12:57:24)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import base64
>>> myPixel = bytearray([255, 0, 0])
>>> myPixel
bytearray(b'\xff\x00\x00')
>>> myArrayBase64 = base64.b64encode(myPixel)
>>> myArrayBase64
'/wAA'
>>> len(myArrayBase64)
4
```

Les Tableaux (arrays)

Outil indispensable pour le traitement de l'information, les tableaux sont omniprésents. Que ce soit une image (tableau d'octets), une chaîne de caractères "Bonjour", ou une liste d'objets, nous ne pouvons les contourner. Plus on travaille avec eux, plus la logique de traitement de masse s'ouvre à nous.

Un fichier texte peut être récupéré en tant que tableau par certains langages. Chaque ligne du tableau correspond alors à une chaîne de caractères du fichier (qui elle-même est un tableau). Le changement de ligne correspond alors à un saut de ligne (caractère '\n' correspondant à la valeur numérique 10 d'un octet).

Passons à un exemple complet utilisant des objets traités dans des tableaux.

Vous devez réaliser deux opérations pour votre projet. L'une permet d'envoyer l'ordre d'extinction d'une lampe à un robot, l'autre permet de mettre en position zéro un tapis roulant.

Dans un premier temps, vous allez écrire un algorithme de ce type:

```
Start
    SwitchOff(Light1)
    ResetEngine(Treadmill1)
End
```

Cependant, au fur et à mesure de l'avancée du projet, vous vous apercevez que vous avez une quantité énorme de lampes à gérer, ainsi que de tapis roulants.

Lors de l'initialisation du projet, après 2 ans de travail, vous avez ajouté dans votre méthode d'initialisation toutes les lampes et tous les tapis roulants les uns à la suite des autres.

```
Function Init()
Start
    # part 1
    SwitchOff(Light1)
    ResetEngine(Treadmill1)

    # part 2
    SwitchOff(Light2)
    ResetEngine(Treadmill2)
```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
# part 3
SwitchOff (Light3)
ResetEngine (Treadmill3)

[...]

# part 29
SwitchOff (Light29)
ResetEngine (Treadmill29)

End
```

Une fois la quantité de paramètres impossible à gérer, vous passez à la revue de code.

Dans ce deuxième temps, vous allez réduire, de manière prudente ces initialisations, tel que:

```
Function Init()
Start
    For i <- 1 To 29
    Do
        nameOfLight <- "Light" + i
        SwitchOff (Eval (nameOfLight))
        nameOfTreadmill <- "Treadmill" + i
        ResetEngine (Eval (nameOfTreadmill))
    Next
End
```

Cela est parfait, jusqu'au moment où votre chef de productique vous demande de ne pas toucher au tapis numéro 15.

Alors, là, par manque de temps, vous allez introduire votre première exception.

```
Function Init()
Start
    For i <- 1 To 29
    Do
        nameOfLight <- "Light" + i
        SwitchOff (Eval (nameOfLight))
        If i = 15
        Then
            Next
        End If
        nameOfTreadmill <- "Treadmill" + i
        ResetEngine (Eval (nameOfTreadmill))
    Next
End
```

Magnifique, livré en temps record avec peu de modifications.

La semaine suivante, c'est la lampe 8 qu'il ne faut plus éteindre au démarrage de l'application, puis la lampe 12, puis le tapis roulant 18, et encore le couple 21.

Les deux premières modifications ont été faites comme la précédente, en exception. Afin de ne pas pourrir un code qui doit tenir des années, vous allez arrêter d'ajouter des rustines dans le code, et le refactoriser.

Deux solutions s'offrent alors à vous:

- revenir dans le tout premier développement (tout écrire sans boucle)
- créer un tableau de paramétrage

Développons notre tableau de paramétrage dans la troisième étape.

Ce tableau contient une ligne par sous-ensemble de la chaîne de production.

Chaque ligne contient elle-même un tableau avec les éléments à traiter.

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
# Define
ParametersArray <-
[
    ["Light1", "Treadmill1"],
    ["Light2", "Treadmill2"],
    ["Light3", "Treadmill3"],
    ["Light4", "Treadmill4"],
    ["Light5", "Treadmill5"],
    ["Light6", "Treadmill6"],
    ["Light7", "Treadmill7"],
    [Nothing, "Treadmill8"],
    ["Light9", "Treadmill9"],
    ["Light10", "Treadmill10"],
    ["Light11", "Treadmill11"],
    [Nothing, "Treadmill12"],
    ["Light13", "Treadmill13"],
    ["Light14", "Treadmill14"],
    ["Light15", Nothing],
    ["Light16", "Treadmill16"],
    ["Light17", "Treadmill17"],
    ["Light18", Nothing],
    ["Light19", "Treadmill19"],
    ["Light20", "Treadmill20"],
    [Nothing, Nothing],
    ["Light22", "Treadmill22"],
    ["Light23", "Treadmill23"],
    [...]
    ["Light29", "Treadmill29"]
]
# Treatment
Function Init()
Start
    For i <- 0 To Count(ParametersArray) - 1
        Do
            nameOfLight <- ParametersArray(i)(0)
            If nameOfLight <> Nothing
                Then
                    SwitchOff(Eval(nameOfLight))
                End If
            nameOfTreadmill <- ParametersArray(i)(1)
            If nameOfTreadmill <> Nothing
                Then
                    ResetEngine(Eval(nameOfTreadmill))
                End If
        Next
    End
```

Cette modularité permet, par la suite, de rajouter des lignes dans le tableau de paramétrage sans pour autant toucher à la boucle et aux traitements liés.

Cela pourrait aller encore plus loin, si l'on cumule le jeu de tableaux avec la généralisation (ou encore l'interface).

On s'aperçoit, dans la boucle ci-dessus, qu'il y a une redondance de code, ce qui n'est pas forcément facile à maintenir quand on a plusieurs centaines de ligne métier.

Si l'on transforme chaque ligne de notre tableau de paramétrage en un tableau d'objets, on pourrait alors définir une interface pour chaque objet qui contrôle les actions à faire par objet.

Par exemple, définissons une interface nommée "ICommands", telle que:

```
Interface ICommands
    Abstract Function Initialize()
    Abstract Function ExecuteProcess()
End
```

Nous pourrions alors créer des classes "Light" et "Treadmill", définissant l'interface

```
Class Light implements ICommand
    name: string

    Function Initialize()
        SwitchOff()
    End

    Function ExecuteProcess()
        DoSomething()
    End

    Function SwitchOff()
        [...]
    End
End
```


Analysez (vos besoins), Etudiez, Scriptez L'art d'utiliser du scripting pour un projet

```
Class Treadmill implements ICommand
    name: string

    Function Initialize()
        ResetEngine()
    End

    Function ExecuteProcess()
        DoSometing(WithParameters)
    End

    Function ResetEngine()
        [...]
    End
End
```

Nous pouvons ainsi redéfinir notre tableau de paramètres en y rajoutant des caractéristiques métier.

```
# Define
ParametersArray <-
    [
        (ICommand) new Light("Light1"),
        (ICommand) new Treadmill("Treadmill1"),
        (ICommand) new Light("Light2"),
        (ICommand) new Treadmill("Treadmill2"),
    [...]]
        (ICommand) new Light("Light29"),
        (ICommand) new Treadmill("Treadmill29"),
    ]
```

Le traitement se fera alors de la manière suivante:

```
# Treatment
Function Init()
    Start
        For i <- 0 To Count(ParametersArray) - 1
            Do
                objectInArray <- ParametersArray(i)
                objectInArray->Initialize()
            Next
    End
```

On peut alors constater que la boucle de traitement est complètement abstraite et ne contient aucune exception.

Les exceptions sont définies à la création des objets.

Pour ma part, c'est ce qui me permet, en industrie, de ne pas faire d'erreur dans une boucle de traitement. Au pire, si un composant ne répond pas tel que l'on voulait, il faut s'intéresser à sa définition, sa création et son comportement, mais le reste globalisé de la chaîne n'est pas impacté.

Ce type de programmation se fait par itérations. Nous en avons trois dans l'exemple, mais cela peut demander plusieurs mois de programmation ou refactorisation lors de gros projets.

Jouer avec les tableaux est donc tout à fait important. Ma philosophie est complètement orientée sur ce modèle et cette façon de penser.

Un exemple concret est la création de plusieurs machines virtuelles dans l'Azure avec PowerShell

```
Import-Module azure

# Connect to the Azure Account
Add-AzureAccount
#Get-AzureSubscription -Current

# Select the "Windows Server 2012 R2 Datacenter" on the catalog list
$ImageName = (
    Get-AzureVMImage |
    Where { $_.ImageFamily -eq "Windows Server 2012 R2 Datacenter" } |
    Sort PublishedDate -Descending |
    Select-Object -First 1).ImageName

# Define the parameters for all subnet machines
$serviceName      = "AZTeamFoundation"
$loginAdmin        = "admin"
$pwdAdmin          = "1234"
$vnetName          = "Azure-VNet01"
$subnetName        = "Subnet-1"
$azureVNet         = Get-AzureVNetSite | ? { $_.Name -like $vnetName }

# Define the configuration array of machines
$vmconfigList = @()
$vmconfigList += @{ vmName="AZTFS02"; vmIP = "*.*.*.*" ; vmSize =
"ExtraSmall" } # TFS (second)
$vmconfigList += @{ vmName="AZTFB01"; vmIP = "*.*.*.*" ; vmSize =
"Small" } # Build
```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
$vmconfigList += @{ vmName="AZTFD01"; vmIP = "*. *.*.*" ; vmSize =
"ExtraSmall" } # Deployment
$vmconfigList += @{ vmName="AZTFM01"; vmIP = "*. *.*.*" ; vmSize =
"ExtraSmall" } # Monitor
$vmconfigList += @{ vmName="AZTFT01"; vmIP = "*. *.*.*" ; vmSize =
"Medium" } # Test
$vmconfigList += @{ vmName="AZTFB02"; vmIP = "*. *.*.*" ; vmSize =
"Small" } # Build for Fast

# Get the Azure Storage account
$azstorage = Get-AzureStorageKey -StorageAccountName "aztfstorage01"

# Get certificate and subscription id
$cert = Get-AzureAccount | Where-Object { $_.Type -like
"Certificate" }
$subid = $cert.Subscriptions

$vmList = @()

function CreateVM {
    Param (
        [Parameter(Mandatory=$True)]
        [String]$vmName,
        [Parameter(Mandatory=$True)]
        [String]$vmIP,
        [String]$vmSize = "Medium"
    )

    $vmTmp = New-AzureVMConfig -Name $vmName -InstanceSize $vmSize
        -ImageName $ImageName |
        Add-AzureProvisioningConfig -Windows
            -AdminUsername $loginAdmin
            -Password $pwdAdmin |
        Set-AzureSubnet -SubnetNames $subnetName |
        Set-AzureStaticVNetIP -IPAddress $vmIP

    return $vmTmp
}

$vmconfigList | ForEach-Object {
    $vmList += ,(CreateVM -vmName $_["vmName"]
                        -vmIP $_["vmIP"] -vmSize $_["vmSize"])
}

# Show the collection
```

```
$vmList | Format-List  
  
# Create the VMs  
New-AzureVM -ServiceName $serviceName -VMs $vmList  
            -VNetName $vnetName -WaitForBoot
```

Pour ce script, tout tourne autour d'un tableau de configuration de machines virtuelles "vmconfigList" qui va définir le nom, l'adresse IP principale et la taille de chaque machine virtuelle.

La philosophie étant posée, nous allons parcourir quelques langages de script, en s'appuyant sur des exemples, pour montrer qu'il est assez aisé de scripter un batch de traitement et pourquoi pas aller même à scripter un client lourd complet.

Différents langages de scripting

Les six chapitres seront dédiés à la présentation de six langages de programmation (PHP, Python, JavaScript, PowerShell, Perl et F#).

Pour chacun d'entre eux, nous allons nous poser la question de "Pourquoi utiliser ce langage", puis nous afficherons le fameux "Hello World !". Ensuite, nous scripterons un exercice en mode console, nous coderons un second exemple plus approfondi, puis nous finirons par voir l'interaction graphique avec ce langage.

Certains se feront la réflexion du langage JavaScript au milieu des autres, qui est un langage orienté client et graphique. Cela pourrait, dans le plan défini, être un contresens même à sa création. Mais justement, c'est parce que nous allons nous intéresser au langage même que l'exercice de le faire rentrer "à l'envers" dans le plan sera intéressant.

PHP

Contenu de ce chapitre

| | |
|--|----|
| PHP..... | 31 |
| Pourquoi utiliser ce langage..... | 32 |
| Afficher simplement 'Hello World'..... | 33 |
| Comment utiliser les scripts par la console..... | 34 |
| Serveur Web en script PHP..... | 34 |
| Aller plus loin avec le scripting en mode console..... | 41 |
| Interaction entre le Scripting et une interface graphique..... | 50 |
| Gestion de la plate-forme hôte..... | 52 |
| Exemple avec le programme « Picture Resizing »..... | 53 |

Pourquoi utiliser ce langage

PHP est un langage magnifique, souvent critiqué par les "vrais" développeurs, le relayant à de la programmation pour débutants.

Le souci est que finalement, oui, nous pouvons développer de manière tout à fait sale, mais nous avons aussi la possibilité d'activer des modes de développement strict, en mettant tout warning comme erreur, par exemple (comme C#). C'est là que l'on s'aperçoit que les "vrais" développeurs ne sont que des ignorants et incapables de s'ouvrir à d'autres mondes que le leur. Généralement, ces personnes se sont mises dans un langage (Java, C#, Ruby), et ne voient plus que ce dernier.

Si l'on suit les recommandations du langage, et que l'on respecte les orientations des créateurs (fonction dépréciée, utiliser plutôt cette dernière), on s'aperçoit que la maintenance est assez simple et ne nécessite que peu de temps.

Le souci dans ce cas est de bien structurer nos besoins dès le départ, et essayer de refactoriser petit à petit, à chaque ajout de fonctionnalité dans le programme.

De plus, le langage PHP est utilisé la plupart du temps pour des applications Web, ce qui représente 70% d'utilisation des fonctions de base du langage. PHP est bien plus que cela et permet de coder et scripter d'autres types d'applications que des interfaces Web. Le jeu avec ses tableaux type 'Dictionnaires' est extrêmement puissant. La communauté derrière ce langage est très développée. Malgré la quantité de mises à jour sur ce moteur, les applications développées il y a 10 ans avec les exigences et les Best Practices sont encore opérationnelles aujourd'hui. Cela n'est pas le cas de tous les langages !

Afficher simplement 'Hello World'

Hello World en PHP par commande ligne se fait de manière très simple.

```
<?php
// fichier hello.php
print("Hello World !\n");
?>
```

```
PS C:\Users\Pierre> php -r "print('Hello World!');"
Hello World!
PS C:\Users\Pierre>
```

```
PS C:\Users\Pierre> php hello.php
Hello World!
PS C:\Users\Pierre>
```

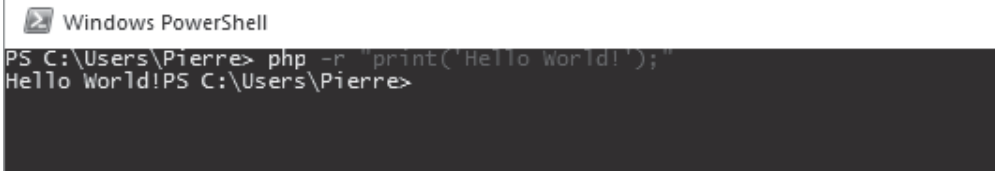


Illustration 1: Hello World en mode console avec PHP

Comment utiliser les scripts par la console

Soit la commande est lancée en direct, soit cette dernière est insérée dans un fichier et celui-ci est transmis à l'interpréteur dans l'invite de commande.

Une bonne entrée en matière pour ceux qui n'utilisent le PHP que pour le web. Ces personnes passent à côté d'une puissance de calcul et d'interprétation en ligne de commande. Elles oublient aussi que l'on peut interfacer PHP avec une interface graphique pour en faire une application en client lourd (cf § Interaction entre le Scripting et une interface graphique).

Serveur Web en script PHP

Voici un petit exemple de client lourd. Le bout de code ci-dessous permet de servir des pages web http très simple, ainsi que d'interpréter au minimum quelques instructions PHP pour les retranscrire. Cet exemple est utilisé dans mon serveur web (Raspberry Pi) qui héberge mes sites web.

```
<?php
error_reporting(E_ALL);

/* Allow the script to hang around waiting for connections. */
set_time_limit(0);

/* Turn on implicit output flushing so we see what we're getting
 * as it comes in. */
ob_implicit_flush();

server_loop("localhost", 8084);
$tabSessions = array();

/**
 * Creates a server socket and listens for incoming client
connections
 * @param string $address The address to listen on
```

```

* @param int $port The port to listen on
*/
function server_loop($address, $port)
{
    global $tabSessions;

    $sock = socket_create(AF_INET, SOCK_STREAM, 0)
        or die("Failed to create socket !\n");

    $ret = socket_bind($sock, $address, $port)
        or die("Failed to bind socket !\n");

    $ret = socket_listen($sock, 0)
        or die("Failed to listen to socket !\n");

    print("Pending connection for clients\n");

    while (true)
    {
        $connection = @socket_accept($sock)
            or die("Error in socket acceptance !\n");

        $quiting = interact($connection);
        socket_close($connection);
        if($quiting) break;
    }
    socket_close($sock);
    print("Bye !\n");
    print_r($tabSessions);
}

```

Voici la fonction de traitement, une fois le client connecté:

```

function interact($socket)
{
    global $tabSessions;

    $InputData = "";
    $OutputStream = "";
    $tabReturn = array();
    $sid = "";

    $scanPrint = true;

    if(socket_recv($socket, $InputData, 4096, 0))
        print($InputData);
}

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
$tabData = explode("\n", $InputData);

$protocolRqt = "";
$askRqt = "";
$codeRqt = "HTTP/1.1";
// get protocol, page and http version
$tabRequest = explode(" ", $tabData[0]);
if(count($tabRequest) > 0) $protocolRqt = $tabRequest[0];
if(count($tabRequest) > 1) $askRqt = $tabRequest[1];
if(count($tabRequest) > 2) $codeRqt = trim($tabRequest[2]);

// get header information between browser and server
$tabHeader = getHeader($tabData);

// default
if($askRqt == "/") $askRqt = "/index.html";

// split parametter and file name
$tabAskRqt = explode("?", $askRqt, 2);
$pageName = $tabAskRqt[0];

// get url parameters
$_GET = array();
if(count($tabAskRqt) > 1)
    $_GET = explode("&", $tabAskRqt[1]);
if(count($_GET))
    splitKeyValue($_GET, '=');

// get post parameters
$_POST = array();
if($protocolRqt == "POST")
{
    $parameters = $tabData[count($tabData) - 1];
    $_POST = explode("&", $parameters);
    if(count($_POST))
        splitKeyValue($_POST, '=');
}
// merge the two parameters
$_REQUEST = array_merge($_GET, $_POST);

if(($protocolRqt == "GET" || $protocolRqt == "POST")
    && file_exists("." . $pageName))
{
    $tabReturn[] = $codeRqt . " 200 OK";
    $tabReturn[] = "Location: http://localhost:8080/";
    $tabReturn[] = date("r");
}
```

Comment utiliser les scripts par la console

```

$tabReturn[] = "Cache-Control: private";
$tabReturn[] = "Server: CPierreWS";

$expire = time() + 3600;
// create or get SID (cookie & session),
// create a cookie or use the existing one
if(isset($tabHeader["Cookie"]))
{
    $sid = trim($tabHeader["Cookie"]);
    $tabReturn[] = "Cookie: " . $sid;
}
else
{
    $sid = "SID:ANON:localhost:" .
        base64_encode(((double) rand()) * 1000000);
    // optional (. ":" . $session_count);
    $tabReturn[] = "Set-cookie: " . $sid;
}

$tabReturn[] = "WWW-Authenticate: Session, "
    . "realm=localhost";
$tabReturn[] = "Allow: GET, HEADER, POST, PUT";

if(strpos($pageName, ".php") !== false)
{
    include(substr($pageName,1));
    $tabReturn[] = "Content-Type: text/html; "
        . "charset=UTF-8";
}
elseif(strpos($pageName, ".js") !== false
    || strpos($pageName, ".htm") !== false)
{
    // include javascript or html page
    $OutputStream = implode("",file(".".$pageName));
    $tabReturn[] = "Content-Type: text/html; "
        + "charset=UTF-8";
}
elseif(strpos($pageName, ".jpg") !== false
    || strpos($pageName, ".bmp") !== false
    || strpos($pageName, ".ico") !== false
    || strpos($pageName, ".gif") !== false)
{
    $imageFic = fopen("." . $pageName, "rb");
    if($imageFic)
    {
        $contents = "";
    }
}

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
        while (!feof($imageFic))
            $contents .= fread($imageFic, 8192);
        fclose($imageFic);
    }
    $OutputStream = $contents;
    $tabReturn[] = "Content-Type: image/"
        . substr($pageName, strlen($pageName) -3, 3);
    $scanPrint = false;
}

$tabReturn[] = "Content-Length: "
    . strlen($OutputStream);
$tabReturn[] = "";
$tabReturn[] = $OutputStream;
}
elseif(strtolower($InputData) == "bye"
    || $pageName == "/bye")
    $tabReturn[] = "Stop serveur : " . date("r");
elseif(!file_exists("." . $pageName))
    $tabReturn[] = "HTTP/1.0 404 Not Found";
else
    $tabReturn[] = "HTTP/1.0 400 Bad Request";

// concatain data array into a string
$returnTxt = implode("\n", $tabReturn);
if(socket_send($socket,$returnTxt,strlen($returnTxt),0))
{
    if($scanPrint)
        print("\n{$returnTxt}\n");
    print("OK data returned\n\n\n");
}
else
    print("\nNOK data not returned\n\n\n");

// save datas into memory for the life time of running server
if($sid != "") $tabSessions[$sid] = $_REQUEST;

// return if ask to stop server
return(strtolower($InputData) == "bye"
    || strtolower($askRqt) == "/bye");
}
```

Vient ensuite une fonction dite "Helper" pour récupérer de manière plus facile les en-têtes transmises par le client:

```
function getHeader($tabSearch)
{
    if(count($tabSearch) < 1)
        return false;

    $tmpArr = array();
    reset($tabSearch);

    // skip the first line (protocole get, post, ..., parameters)
    next($tabSearch);
    // browse the next line until empty line
    while($strTmp = next($tabSearch)) {
        if($strTmp == "")
            break;
        if(strpos($strTmp,":") !== false) {
            $tmpTab = explode(":", $strTmp, 2);
            $tmpArr[$tmpTab[0]] = trim($tmpTab[1]);
        }
    }
    return $tmpArr;
}
```

Une dernière fonction, destinée à séparer les clés-valeurs passées en paramètre au serveur:

```
function splitKeyValue(&$tabToSplit, $splitCaract)
{
    if(count($tabToSplit) <= 0)
        return false;

    $tmpArr = array();
    foreach($tabToSplit as $value) {
        $tmpTab = explode($splitCaract, $value);
        if(count($tmpTab) > 1)
            $tmpArr[$tmpTab[0]] = urldecode($tmpTab[1]);
    }
    $tabToSplit = $tmpArr;
    return true;
}
?>
```


Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

Le résultat en image :

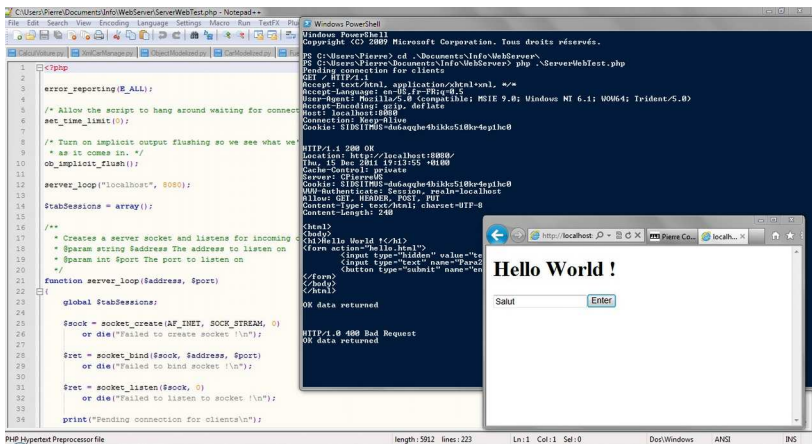


Illustration 2: Le serveur web en action

Aller plus loin avec le scripting en mode console

Imaginons un site web, équipé d'un CMS, qui enregistre chaque page consultée par les utilisateurs. Nous pourrions recenser alors l'adresse IP, la date / heure, le site choisi, la page dans le site.

Ce mini projet, nous permettra d'avoir des statistiques sur un site Web.

Ex:

Fichier de log récupéré à chaque requête web reçue sur le serveur

| DateTime | IP Address | Subsite visited | Category selected |
|----------------|-----------------|---|-------------------|
| 02/11/14 08:42 | xxx.xxx.xxx.xxx | ./data/covering/ covering_en.php | autresmorceaux |
| 02/11/14 08:47 | xxx.xxx.xxx.xxx | ./data/cv/cv.php | curriculumvitae |
| 02/11/14 12:21 | xxx.xxx.xxx.xxx | ./data/instrumental/ instrumental_en.php | presentation |
| ... | ... | ... | ... |

Considérons ces enregistrements comme une liste d'objets contenant quatre champs. Créons donc une classe nommée "LogInfo"

```
class LogInfo {  
    public $dt          = 0;  
    public $ipAddr      = "";  
    public $page        = "";  
    public $category    = "";  
[...]  
}
```

Ajoutons lui un champ supplémentaire 'countryName' contenant le nom du pays lié à l'adresse IP récupérée pendant la visite du site. Ce champ sera calculé dans un second temps.

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

Surchargeons ensuite le constructeur pour lui donner un tableau de 4 éléments correspondant au tableau de logs, et ré-écrivons la fonction d'affichage de l'objet.

```
class LogInfo {

    public $dt          = 0;
    public $ipAddr      = "";
    public $page        = "";
    public $category    = "";
    public $countryName = "";

    public function __construct($tmpArray) {
        if(is_array($tmpArray) && count($tmpArray) > 3) {
            $this->dt          = $tmpArray[0];
            $this->ipAddr      = $tmpArray[1];
            $this->page        = $tmpArray[2];
            $this->category    = $tmpArray[3];

            $this->countryName =
                GlobalStats::getCountryName($this->ipAddr);
        }
    }

    public function __toString() {
        return "Date : {$this->dt}; IPAddress : {$this->ipAddr}; " .
            "Page : {$this->page}; Category : {$this->category}; " .
            "Pays : {$this->countryName}";
    }
}
```

Cette classe doit ensuite être utilisée dans un processus qui lit chaque ligne du fichier de log, construit un objet par ligne et calcule le nom du pays par rapport à une liste de plages d'adresses IP définie par pays.

Créons la classe de gestion des informations IP par pays.

```
class CountryInfo {
    private $IPStart = 0;
    private $IPEnd   = 0;
    private $A2      = "";
    private $A3      = "";
    private $Name     = "";

    public function __construct($tmpLine) {
        if(is_array($tmpLine) && count($tmpLine) > 4) {
            $countTmpLine = count($tmpLine);
            $this->IPStart = floatval($tmpLine[0]);
            $this->IPEnd   = floatval($tmpLine[1]);
            $this->A2      = $tmpLine[2];
            $this->A3      = $tmpLine[3];
            $this->Name     = $tmpLine[4];
        }
    }

    public function __toString() {
        return
            "IPStart : {$this->IPStart}; IPEnd : {$this->IPEnd}; " .
            "A2 : {$this->A2}; A3 : {$this->A3}; Name : {$this->Name}";
    }

    public function containtIP($lIPAddr) {
        return ($this->IPStart <= $lIPAddr &&
            $this->IPEnd >= $lIPAddr);
    }

    public function getCountryName() {
        return ($this->Name != "") ? $this->Name : $this->A2;
    }
}
```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

Créons une classe de calcul des statistiques comprenant le tableau de référencement des pages d'adresses IP par rapport aux noms des pays, les fonctions de conversion d'adresse IP entre flottant et chaîne de caractères, ainsi que le tableau des statistiques de pages consultées.

```
class GlobalStats {
    public static $countryInfoList = array();
    public static $ipList = array();
    public static $pageList = array();
    public static $categoryList = array();
    public static $countryNameCount = array();
    public static $countryNameDetails = array();

    public static function sumInfos($tmpLogInfo) {
        if(!isset(self::$countryNameCount
                    [$tmpLogInfo->countryName]))
            self::$countryNameCount[$tmpLogInfo->countryName] = 1;
        else
            self::$countryNameCount[$tmpLogInfo->countryName]++;

        if(!isset(self::$pageList[$tmpLogInfo->page]))
            self::$pageList[$tmpLogInfo->page] = 1;
        else
            self::$pageList[$tmpLogInfo->page]++;

        if(!isset(self::$categoryList[$tmpLogInfo->category]))
            self::$categoryList[$tmpLogInfo->category] = 1;
        else
            self::$categoryList[$tmpLogInfo->category]++;

        if(!isset(self::$countryNameDetails
                    [$tmpLogInfo->countryName][$tmpLogInfo->ipAddr]))
            self::$countryNameDetails
                [$tmpLogInfo->countryName][$tmpLogInfo->ipAddr] = 1;
        else
            self::$countryNameDetails
                [$tmpLogInfo->countryName][$tmpLogInfo->ipAddr]++;
    }
}
```

Aller plus loin avec le scripting en mode console

```

public static function getCountryName($tmpIpAddr) {
    $tmpIpAddr = self::IPAddress2IPNumber($tmpIpAddr);

    if(!isset(self::$ipList[$tmpIpAddr])) {
        self::$ipList[$tmpIpAddr] = "";
        foreach(self::$countryInfoList as $countryLine) {
            if($countryLine->containtIP($tmpIpAddr)) {
                self::$ipList[$tmpIpAddr] =
                    $countryLine->getCountryName();
                break;
            }
        }
    }
    return self::$ipList[$tmpIpAddr];
}

```

```

public static function loadArrayOfCountry($fileCountry) {
    self::$countryInfoList = array();
    $ficCountry = fopen($fileCountry, "r")
        or die("Erreur in opening country file !\n");
    if($ficCountry != null) {
        while($countryLine = fgetcsv($ficCountry, 150, ",", "\"")) {
            self::$countryInfoList[] = new CountryInfo($countryLine);
        }
        fclose($ficCountry);
    }
}

public static function IPAddress2IPNumber($dotted) {
    $dotted = preg_split( "/[.]+/", $dotted);
    return (double) ($dotted[0]*0x1000000)+($dotted[1]*0x10000)+
        ($dotted[2]*0x100)+($dotted[3]);
}

public static function IPNumber2IPAddress($number) {
    $a = ($number/0x1000000)%0x100;
    $b = ($number/0x10000)%0x100;
    $c = ($number/0x100)%0x100;
    $d = ($number)%0x100;
    $dotted = $a.".".$b.".".$c.".".$d;
    return $dotted;
}
} // End of GlobalStats Class

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
function getArrayOfLog($fileLog) {
    $arrayLogs = array();
    $ficAnalyse = fopen($fileLog, "r")
                    or die("Erreur in opening log file !\n");
    if($ficAnalyse != null) {
        while($data = fgetcsv($ficAnalyse, 200, ";")) {
            $infoLogLine = new LogInfo($data);
            GlobalStats::sumInfos($infoLogLine);
            $arrayLogs[] = $infoLogLine;
        }
        fclose($ficAnalyse);
    }
    return $arrayLogs;
}
```

Ensuite, il ne reste plus qu'à ajouter un Main au programme.

```
// -----
// MAIN

// management on the bash starting
// "/?"; "--help"; "-h" or "yyyyMM" of log statistics
$dateLogs = (count($argv) > 1)?$argv[1]:"all";
if($dateLogs == "--help" || $dateLogs == "/"? || $dateLogs == "-h")
{
    print("Aide sur StatLog\n");
    print("=====\n");
    print("./statLog.php [date yyyyMM] [directoryLogs]\n");
    return 0;
}
else if(!(is_numeric($dateLogs) || $dateLogs == "all")) {
    print("Erreur du format de la date : yyyyMM\n");
    return -1;
}

// 2nd argument is the path of directory logs "default == ./logs"
$directoryLogs = (count($argv) > 2)?$argv[2]:"./logs";
if(!file_exists($directoryLogs)) {
    print("Erreur d'ouverture du repertoire {$directoryLogs} !\n");
    return -1;
}

$startTime = microtime(true);

print("Statistiques des connections sur
Free\n=====\n\n");
// get file about defined of ip country
GlobalStats::loadArrayOfCountry("./ip-to-country.csv");

$timeLoadCountry = microtime(true);

// get array of ip to analyse
$logArray = array();
if($dateLogs == "all") {
    // get all file logs in directory
    // open this directory
    $myDirectory = opendir($directoryLogs);
    // get each entry
    $arraysLogsTmp = array();
    while($entryName = readdir($myDirectory)) {
        if(strpos($entryName, ".log")) {
            $arrayLogsTmp = getArrayOfLog("{ $directoryLogs }"
                . "/" . $entryName);
        }
    }
}
```


Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
        $logsArray = array_merge($logsArray, $arrayLogsTmp);
    }
}
// close directory
closedir($myDirectory);
}
else {
    $logsArray =
        getArrayOfLog("{ $directoryLogs }/logConnexion{$dateLogs}.log");
}

$timeLoadFile = microtime(true);

// count the visiting times for each pages

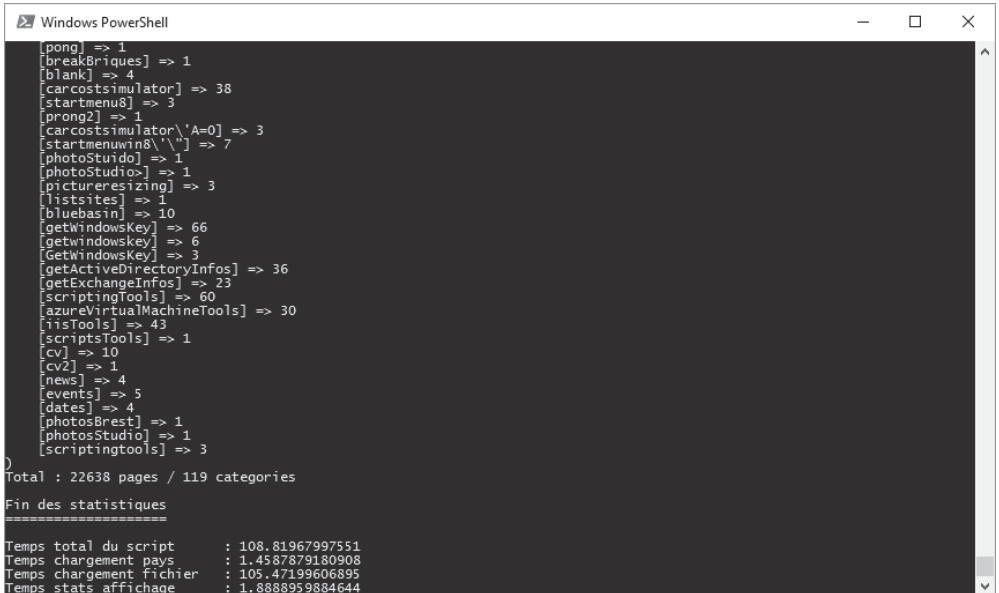
$countNbVisitor = 0;
foreach(GlobalStats::$countryNameDetails as $tmpArr)
    $countNbVisitor += count($tmpArr);

print("Detail des adresses par pays :\n");
print_r(GlobalStats::$countryNameDetails);
print("Nombre de visiteurs : {$countNbVisitor}\n");
print("Resume des adresses par pays :\n");
print_r(GlobalStats::$countryNameCount);
print("Total : ");
print(array_sum(array_values(GlobalStats::$countryNameCount)));
print("\n\n");
print("Nombre de visites par pages :\n");
print_r(GlobalStats::$categoryList);
print("Total : ");
print(array_sum(GlobalStats::$categoryList));
print(" pages / " . count(GlobalStats::$categoryList));
print(" categories\n\n");
print("Fin des statistiques\n=====\n\n");

// end of script
$endTime = microtime(true);

print("Temps total du script      : " . ($endTime - $startTime) .
"\n");
print("Temps chargement pays      : " . ($timeLoadCountry -
$startTime) . "\n");
print("Temps chargement fichier   : " . ($timeLoadFile -
$timeLoadCountry) . "\n");
print("Temps stats affichage      : " . ($endTime - $timeLoadFile) .
"\n");
```

Une partie du résultat en impression d'écran ci-dessous:



```
Windows PowerShell

[pong] => 1
[breakBriques] => 1
[blank] => 4
[carcostimulator] => 38
[startmenu8] => 3
[prong2] => 1
[carcostimulator\A=0] => 3
[startmenuwin8\'] => 7
[photoStudio] => 1
[photoStudio] => 1
[pictureresizing] => 3
[listsites] => 1
[bluebasin] => 10
[getWindowsKey] => 66
[getWindowsKey] => 6
[GetWindowsKey] => 3
[getActiveDirectoryInfos] => 36
[getExchangeInfos] => 23
[scriptingTools] => 60
[azureVirtualMachineTools] => 30
[isTools] => 43
[scriptsTools] => 1
[cv] => 10
[cv2] => 1
[news] => 4
[events] => 5
[dates] => 4
[photosBrest] => 1
[photosStudio] => 1
[scriptingtools] => 3
)
Total : 22638 pages / 119 categories

Fin des statistiques
=====
Temps total du script      : 108.81967997551
Temps chargement pays     : 1.4587879180908
Temps chargement fichier  : 105.47199606895
Temps stats affichage     : 1.8888959884644
```

Illustration 3: Résultat des statistiques

Interaction entre le Scripting et une interface graphique

Prenons un autre exemple: le redimensionnement d'images.

Au début du web, un site devait envoyer de petites images aux clients car les débits ne permettaient pas de télécharger des quantités importantes de données.

La technique consistait donc à afficher des miniatures de photos plutôt que l'image originale. Lorsque l'on souhaite voir une photo en pleine écran, à ce moment l'utilisateur clique sur la miniature et l'appel web va chercher la photo de taille importante.

Sur un site fortement utilisé contenant des centaines de photos par page, il y avait deux solutions pour afficher des miniatures de photos:

- Réduction à la volée.
 - ✓ Avantages: gain de place sur disque dur, adaptation très dynamique lors d'un upload de photos.
 - ✓ Inconvénients: calcul processeur à la volée et à l'appel de chaque page impliquant des lenteurs à l'affichage de la page, nombre d'IO doublé par le calcul à la volée et utilisation de mémoire vive plus importante car le stockage des miniatures se fait soit en mémoire vive, soit dans un dossier temporaire de la machine.
- Réduction complète des images et stockage de ces dernières sur disque dur.
 - ✓ Avantages: rapide, les photos miniatures n'ont pas besoin d'être recalculées à chaque appel.
 - ✓ Inconvénients: consommation d'espace disque dur important et besoin d'un pré traitement à l'upload d'une photo.

Appliquons l'exemple de redimensionnement d'images avec les prérequis suivants:

- Serveur non clusterisé, non dédié,
- Petit processeur (700MHz)
- Système d'exploitation hôte: Debian
- Hébergement de plusieurs sites web
- Limite de disque dur 10Go
- Taille globale des 8 sites 4Go
- Mémoire vive limitée: 512Go

Avec ces prérequis, la décision a été de créer des miniatures d'images pour les sites et sous-sites qui contiennent des banques d'images à afficher sur une page web. Les sites web étant en majorité développés en PHP, la continuation logique de la gestion des images s'est faite en PHP. De plus, le temps restreint pour réaliser les interfaces, ainsi que la gestion multi OS a orienté le choix PHP de manière logique. Pour rappel, ce script a été réalisé en 2008, mono (utilisation de C# sous Linux) n'était pas encore abouti et Java demandait trop de temps d'analyse des bonnes bibliothèques pour être multi plate-forme. Au niveau de l'interface graphique, le choix s'est porté sur GTK+, car son intégration avec PHP se fait de manière simple.

Plusieurs étapes ont été nécessaires pour traiter les images:

1. Création d'un POC de réduction d'une image à la volée sur un site Bac à Sable
2. Extraction des fonctions de calcul dans une feuille séparée permettant sa réutilisation.
3. Création d'un script PHP appelant le traitement de l'image.
4. Transformation du script pour traiter le contenu d'un dossier complet avec plusieurs images.
5. Ajout de l'interface homme machine pour une utilisation plus facile.

Pour cette partie, je vais introduire le développement de l'interface graphique avec GTK+. Cela est vraiment intéressant grâce à la facilité d'installation et de développement. GTK+ étant multiplate-forme, le programme est ensuite lisible par le plus grand nombre de nos ordinateurs (PC, Linux, OSX) après y avoir installé son moteur de rendu.

Un exemple d'application utilisant GTK+ est le célèbre logiciel GIMP de retouche d'images libre d'utilisation.

Gestion de la plate-forme hôte

- Sur une plate-forme Windows

Avec PHP, vous pouvez télécharger le projet "PHP-GTK" en tant que paquet zip. Dézipper et copier le contenu à l'emplacement "C:\Program files (x86)". Référencer le chemin d'accès dans la variable d'environnement "%PATH%", lancer la console et l'utiliser.

Si vous utilisez PHP comme CGI et PHP-GTK, renommez le fichier "php.exe" du nouveau répertoire installé par "php-gtk.exe". Vous pourrez alors utiliser les deux programmes sans risque de mauvaise interaction.

- Sur plate-forme Linux

Sur une Debian, vous lancez la console en mode "Super Utilisateur", puis tapez "apt-get install build-essential php5-cli php5-dev libgtk2.0-dev libglade2-dev". Même philosophie sur Fedora en utilisant yum "yum install gtk2". Vraiment simple.

Exemple avec le programme « Picture Resizing »

Trois parties dans cet exemple sans Glade:

- Create the compress picture library
- Création de l'IHM
- Liaison entre les modules et l'IHM par le Main

Compress Picture Library

```
<?php

/* CompressImg() */
function compressImg($Image, $TailleX, $tmpInputDirectory = "",
$tmpOutputDirectory = "", $partImgCmp = "cmp_") {
    // On cherche a obtenir les info concernant l'image
    // (repertoire de base, nom et extension)
    $tmpImgName = $tmpInputDirectory . "/" . $Image;
    $InfoFile = pathinfo($tmpInputDirectory . "/" . $Image);
    $Extension = $InfoFile['extension'];

    // Checking extension file
    switch ( strtolower($Extension) ) {
        case "jpg":
            $RessourceImage = imagecreatefromjpeg($tmpImgName);
            break;

        case "jpeg":
            $RessourceImage = imagecreatefromjpeg($tmpImgName);
            break;

        case "gif":
            $RessourceImage = imagecreatefromgif($tmpImgName);
            break;

        case "png":
            $RessourceImage = imagecreatefrompng($tmpImgName);
            break;

        // Unknown extension
        default :
```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
        echo "Erreur! Picture $ImageACompressor is incompatible!";
        exit();
    }

    // Get width and height
    $ImgLargeur= imagesx($RessourceImage);
    $ImgHauteur = imagesy($RessourceImage);

    // ratio calculating
    $Coef = $ImgLargeur/$TailleX;

    // resize height with previous ratio
    $Largeur = $TailleX;
    $Hauteur = round(($ImgHauteur/$Coef ), 0);

    // Create a new empty picture with good format
    $Img = imagecreatetruecolor($Largeur, $Hauteur);

    // Copy the oldest picture into the new empty one
    imagecopyresampled($Img, $RessourceImage, 0, 0, 0, 0, $Largeur,
        $Hauteur, $ImgLargeur, $ImgHauteur);

    // Save the new picture
    imagejpeg($Img, $tmpOutputDirectory . "/" . $partImgCmp . $Image);
}

function content_list($repertoire, $type) {
    $imgTbl = array();
    $d=opendir($repertoire);
    if($d) {
        while($f=readdir($d))
            if(stristr($f,$type)) $imgTbl[] = $f;
        closedir($d);
    }
    return $imgTbl;
}
```

Exemple avec le programme « Picture Resizing »

```
function compressImage($tailleImgX, $repertoireInputTmp = ".",
$repertoireOutputTmp = "") {
    if($repertoireOutputTmp == "")
        $repertoireOutputTmp = $repertoireInputTmp;

    $tblImg = content_list($repertoireInputTmp, ".jpg");
    foreach($tblImg as $tmpImg) {
        compressImg($tmpImg, $tailleImgX, $repertoireInputTmp,
            $repertoireOutputTmp);
    }

    return $tblImg;
}
?>
```

Sans Glade, il suffit de coder en xml l'IHM

```
#!/usr/bin/php
<?php
// picture reductor software
// using GTK

$windows = array();

// include GTK library
//if (!extension_loaded('php-gtk')) {
//    dl( 'php_gtk2.' . PHP_SHLIB_SUFFIX);
//}
if (!class_exists('gtk')) {
    die('Please load the php-gtk2 module in your php.ini' . "\r\n");
}

// include the compress image code
if(file_exists("./compressImage.php"))
    include "./compressImage.php";
else
    die("Erreur de chargement du fichier de compression !\n");

function delete_event($window, $event)
{
    $window->hide();
    return true;
}
```


Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
function close_window($widget)
{
    $window = $widget->get_toplevel();
    $window->hide();
}

function chooseDirectory_event($window, $event = null)
{
    global $windows;
    $windows['tmpEventName'] = $window->get_name();
    $tmpFileName = create_directory_selection();
    $windows[$windows['tmpEventName']->set_text($tmpFileName);
    return true;
}

function create_directory_selection()
{
    $fileNameTmp = "";

    $window = &new GtkFileChooserDialog('Select a directory',
        null, Gtk::FILE_CHOOSER_ACTION_SELECT_FOLDER,
        array(Gtk::STOCK_CANCEL, Gtk::RESPONSE_CANCEL,
            Gtk::STOCK_OPEN, Gtk::RESPONSE_OK));
    $window->set_default_response(Gtk::RESPONSE_OK);
    $response = $window->run();
    if($response == Gtk::RESPONSE_OK)
        $fileNameTmp = $window->get_filename();
    $window->destroy();

    return $fileNameTmp;
}

function reduce_pictures($window, $event = null)
{
    global $windows;

    $dialog = &new GtkDialog;
    $windows['dialog'] = $dialog;
    $dialog->set_title('Compress Image OK');
    $dialog->set_border_width(0);
    $dialog->set_size_request(350, 120);
    $dialog->connect('delete_event', 'delete_event');

    $button = &new GtkButton('Ok');
    $button->set_flags(Gtk::CAN_DEFAULT);
    $button->connect('clicked', 'close_window');
```

Exemple avec le programme « Picture Resizing »

```

        $action_area = $dialog->action_area;
        $action_area->pack_start($button, true, true, 0);
        $button->grab_default();
        $button->show();

        $text_area = $dialog->vbox;
        $label = &new GtkLabel('Merci de patienter '
                                + "pendant la compression des images.");
        $text_area->add($label);
        $label->show();

        $dialog->show();

        $tmpIn = $windows['inputBoxIn']->get_text();
        $tmpOut = $windows['inputBoxOut']->get_text();
        $tmpType = "jpg";

        $lstImg = compressImage(600, $tmpIn, $tmpOut,
                                $tmpType);
        $label->set_text("Nombre d'images compressées : "
                        . count($lstImg) . "\n ==> OK");

        return true;
}

function checkButtonOutput_event($window, $event = null)
{
    global $windows;
    if(!isset($windows['boxOut'])) return false;

    if($window->get_active())
        $windows['boxOut']->hide();
    else
        $windows['boxOut']->show();
}

function main_destroy() {
    Gtk::main_quit();
}

function exit_event() {
    return false;
}

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
//create main windows
function create_main_window()
{
    global $windows;

    $window = &new GtkWindow;
    $windows['main'] = $window;
    $window->set_resizable(false);
    $window->set_name('main window');
    $window->set_size_request(600, 150);
    $window->set_position(GTK_WIN_POS_CENTER);
    $window->set_title('Picture reducing');
    $window->connect('destroy', 'main_destroy');
    $window->connect('delete-event', 'exit_event');

    // add box with content of inputs, and output directory
    $box0 = &new GtkVBox();
    $window->add($box0);

    $box1 = &new GtkHBox();
    $box0->add($box1);

    $separator = &new GtkHSeparator();
    $box0->add($separator);

    $checkButton = &new GtkCheckButton('Same direcotry in output');
    $checkButton->connect('clicked', 'checkButtonOuput_event');
    $box0->add($checkButton);

    $box2 = &new GtkHBox();
    $windows['boxOut'] = $box2;
    $box0->add($box2);

    $separator = &new GtkHSeparator();
    $box0->add($separator);

    $box3 = &new GtkHBox();
    $box0->add($box3);

    // label "In" (box1)
    $labelIn = &new GtkLabel('Enter the directory of input pictures :');
    $box1->add($labelIn);
    // input box "In"
    $inputBoxIn = &new GtkEntry();
    $windows['inputBoxIn'] = $inputBoxIn;
```

Exemple avec le programme « Picture Resizing »

```

$inputBoxIn->set_text('');
$box1->add($inputBoxIn);
// button choose directory
$buttonChooseDirectory = &new GtkButton('...');
$buttonChooseDirectory->set_name('inputBoxIn');
$buttonChooseDirectory->connect('clicked',
                                'chooseDirectory_event');
$box1->add($buttonChooseDirectory);

// label "Out" (box2)
$labelOut = &new GtkLabel('Enter output pictures directory :');
$box2->add($labelOut);
// input box "Out"
$inputBoxOut = &new GtkEntry();
$windows['inputBoxOut'] = $inputBoxOut;
$inputBoxOut->set_text('');
$box2->add($inputBoxOut);
// button choose directory
$buttonChooseDirectory = &new GtkButton('...');
$buttonChooseDirectory->set_name('inputBoxOut');
$buttonChooseDirectory->connect('clicked',
                                'chooseDirectory_event');
$box2->add($buttonChooseDirectory);

// button start (box3)
$buttonStartReduce = &new GtkButton('Start reduce');
$buttonStartReduce->connect('clicked', 'reduce_pictures');
$box3->add($buttonStartReduce);

// button quit (box3)
$buttonQuit = &new GtkButton('Exit');
$buttonQuit->connect('clicked', 'main_destroy');
$box3->add($buttonQuit);

// show main window
$window->show_all();
}

create_main_window();
Gtk::main();
?>

```

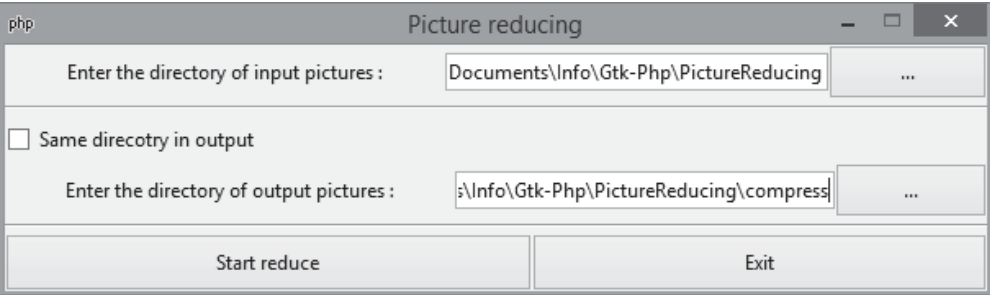


Illustration 4: Ecran principal de PictureResizing sans l'interface Glade

Avec Glade, vous pouvez utiliser une interface graphique pour créer votre propre Interface Homme Machine.

Glade IHM

```
<?xml version="1.0"?>
<glade-interface>
  <!-- interface-requires gtk+ 2.16 -->
  <!-- interface-naming-policy project-wide -->
  <widget class="GtkWindow" id="PictureResizing">
    <property name="visible">True</property>
    <property name="extension_events">all</property>
    <property name="title" translatable="yes">Reduce
picture</property>
    <property name="resizable">False</property>
    <child>
      <widget class="GtkVBox" id="vbox1">
        <property name="visible">True</property>
        <property name="orientation">vertical</property>
        <child>
          <widget class="GtkMenuBar" id="menubar1">
            <property name="visible">True</property>
            <child>
[...]
```

```
          </widget>
        </child>
      </widget>
    </glade-interface>
```

N'ayant aucun intérêt de montrer tout le code ici, vous pouvez le télécharger sur mon site (<http://pierre.contri.free.fr/?site=programming>).

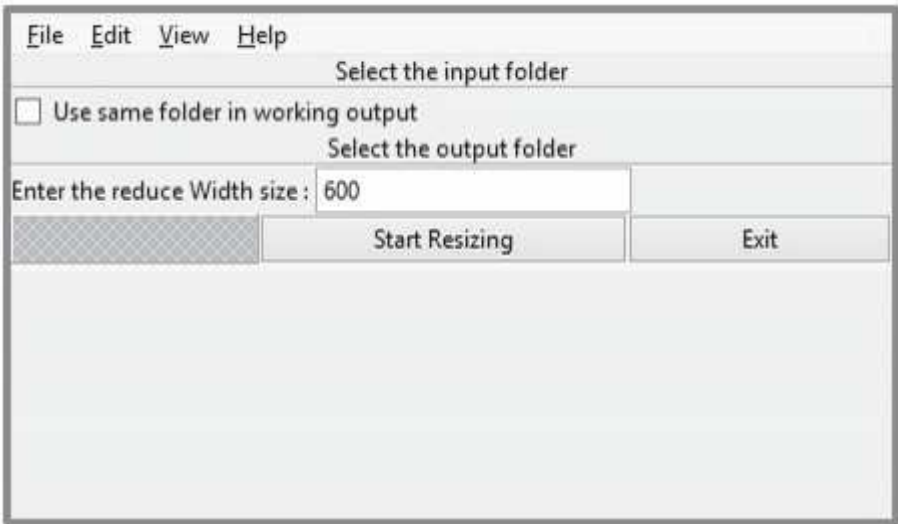


Illustration 5: Ecran principal de PictureResizing avec Glade



Illustration 6: Ecran d'aide de PictureResizing avec Glade

Main program

```
#!/usr/bin/php
<?php
// picture reductor software
function main_destroy() {
    Gtk::main_quit();
}

function exit_event() {
    return false;
}

//Create a new glade instance, load the
// widgets from the file passed as parameter
$glade = new GladeXML(dirname(__FILE__) . '/PictureResizing.glade');

//Nothing happened when you clicked the button or closed
// the window with Step 1's code.
//Here we manually connect the widget signals as you know it
// $window = $glade->get_widget('wndClose');
// $window->connect_simple('destroy', array('Gtk', 'main_quit'));

//Again, get the widget object and connect the clicked signal
// $button = $glade->get_widget('btnClose');
// $button->connect_simple('clicked', 'onClickButton');

//Let glade do all the signal connections we specified in the file
$glade->signal_autoconnect();

$window = $glade->get_widget('PictureResizing');
$window->connect('destroy', 'main_destroy');
```

Par ce script, nous pouvons aisément comprendre que PHP ne doit pas être utilisé que pour le Web. Tant de choses sont possibles avec que ce magnifique langage mérite d'être connu et respecté. Il mérite d'être étudié et non pas de servir de langage d'affichage web uniquement. Le problème de ce langage est sa facilité de prise en main, ainsi que sa gratuité et sa grande communauté.

En outre, ce dernier peut aussi être compilé comme du .Net ou du Java.

Mais ce langage n'est pas le seul à mériter d'être connu. Essayons d'approfondir un peu l'analyse du Scripting en nous diversifiant et en évoquant le Python.

Python

Contenu de ce chapitre

| | |
|--|----|
| Python..... | 65 |
| Pourquoi utiliser ce langage..... | 66 |
| Afficher simplement 'Hello World'..... | 67 |
| Comment utiliser les scripts par la console..... | 68 |
| Aller plus loin avec le Scripting en mode console..... | 69 |
| Interaction entre le Scripting et une interface graphique..... | 85 |
| Utiliser GTK+ et Python..... | 87 |
| Exemple de programme: Car Cost Simulator..... | 87 |

Pourquoi utiliser ce langage

Python est installé par défaut sur les systèmes Linux. Il est compatible avec tous les systèmes d'exploitation et permet, de ce fait, de pouvoir manipuler des fichiers, des données XML, CVS, ou autre de manière très facile. En outre, ce langage est orienté objet.

Ce langage est interprété, mais peut aussi se compiler.

Ce langage peut servir à pas mal de choses bien différentes, telles que:

- créer des scripts de configuration de machines virtuelles pour l'Azure
- servir des pages Web, comme PHP ou Ruby
- automatiser des installations de programme sous Linux
- interfacer des bibliothèques de mathématique (Matlab, Pylab)

Ce langage est riche de modules développés par une communauté libre. Les codes sources sont presque tous ouverts, ce qui permet d'apprendre beaucoup des autres développeurs.

Ce sublime langage a cependant subi des transformations suffisamment importantes pour avoir une scission entre deux versions.

En tant que développeur, je comprends que les langages évoluent et doit accepter le changement de Python, mais clairement, je dois vivre avec deux versions de Python (2.7 et 3.x) car certains modules contenus dans Python 2.6 ou 2.7 n'ont pas été traduits en Python 3.x. J'aimerais traduire un module pour la communauté Python, mais, ce n'est pas avec une heure par semaine que je peux offrir mes services à la communauté.

Afficher simplement 'Hello World'

Tout comme PHP, il est assez timple d'installer Python sous système Windows. Au niveau Linux, il est intégré dans le noyau, ce qui est assez agréable. Après avoir lancé une console, il ne reste plus qu'à taper 'Hello World !'.

```
PS C:\Users\Pierre> python -c "print('Hello World');"
Hello World
PS C:\Users\Pierre>
PS C:\Users\Pierre> python hello.py
Hello World!
PS C:\Users\Pierre>
```

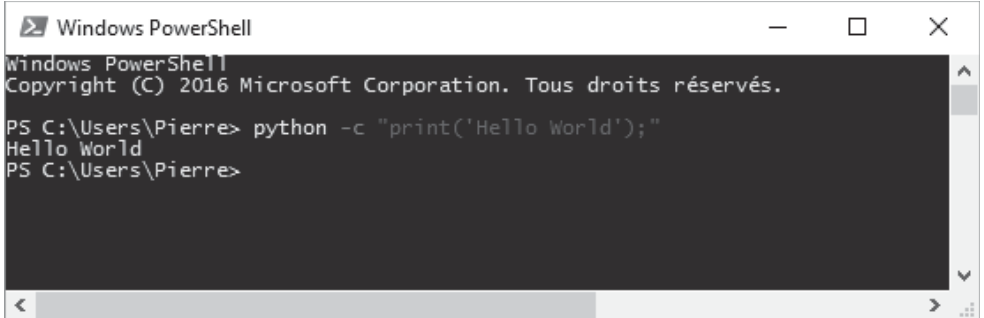


Illustration 7: HelloWorld en PowerShell

Nous pouvons aussi utiliser le mode interactif et ainsi exécuter les ordres en direct sur l'interpréteur.

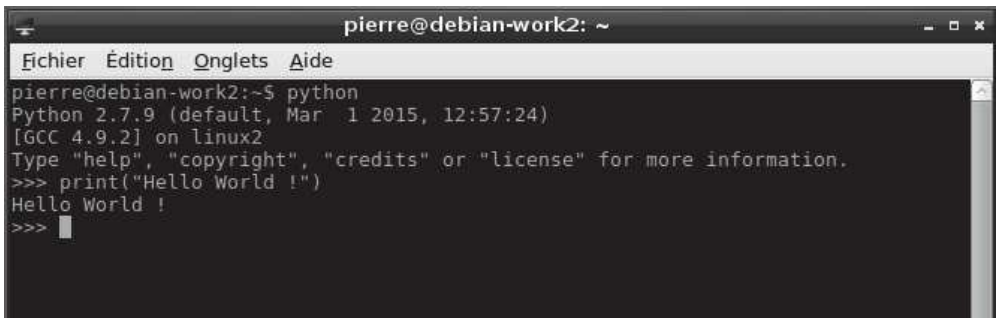


Illustration 8: Interpréteur Python

Comment utiliser les scripts par la console

L'utilisation des scripts par la console se fait de manière très simple comme nous avons pu le voir au précédent paragraphe. Nous avons aussi un outil (IDLE) dans Python permettant de développer des scripts et de les exécuter, ainsi que de les déboguer.

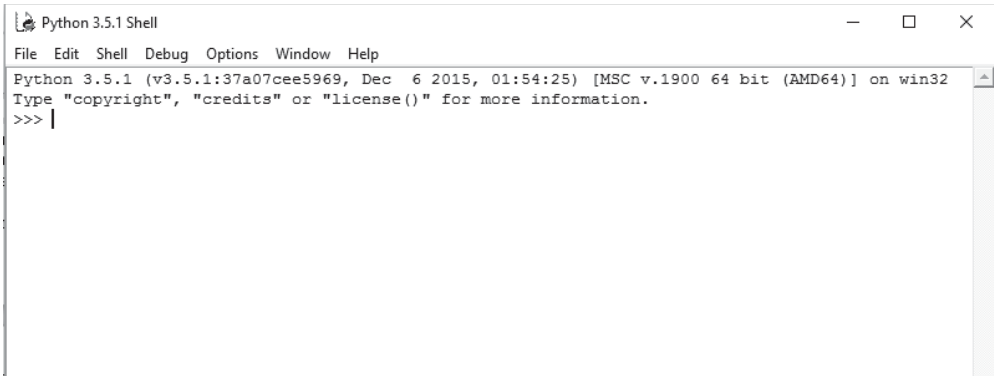


Illustration 9: IDLE (Python GUI)

L'avantage de travailler avec cette console est que l'on est obligé de le faire en Python. Si on lance un script en Python dans une console Linux ou DOS, on a la possibilité d'utiliser le langage bash, sh ou dos pour transformer une chaîne de caractère sortie d'une commande Python. Avec ce shell, c'est vraiment « full Python ».

Aller plus loin avec le Scripting en mode console

Python permet d'écrire un programme de manière modulaire, par fichiers. Certains regroupements de fichiers servent à écrire des modules ou bibliothèques réutilisables.

Reprenons maintenant l'exemple du calcul de statistiques d'un site web déjà traité en PHP.

Nous pourrons aussi voir l'utilisation de la philosophie de la programmation fonctionnelle sur certaines lignes spécifiques.

Comme tout programme moderne, nous utilisons des classes du système pour travailler en haut niveau:

```
#!/usr/bin/env python
import csv, sys, re, math
from os import listdir
from os.path import isfile, join, isdir, abspath
from time import time, strftime, gmtime
from datetime import datetime
from pprint import pprint
from statistics import mean
from pylab import *
import matplotlib.pyplot as plt
import matplotlib as mpl
```

Attaquons l'exercice par la création de la classe Log:

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
class LogInfo(object):
    """
    This object contains informations about each pages
    consulted on the web sites
    with datetime, ip address, page, category
    """

    # the reg exp used to extract a date time
    re_datetime = re.compile(r'^.{0,4} (?P<dt>\d{4} (-\d{2}){2}) ' \
                              ' (?P<tm>\d{2} (: \d{2}){2}) ')

    def __init__(self, parsing_row: list):
        self.dt = ""
        self.ip_addr = ""
        self.page = ""
        self.category = ""
        self._countryname = ""
        self.__try_get_country_name = True

        # parse the input data
        if (len(parsing_row) > 3)
            and IPHelper.is_valide_ip_address(parsing_row[1]):
                self.dt, self.ip_addr, self.page, self.category =
                    parsing_row[:4]

    def search_country_name(self):
        """
        This function is used to find the country name
        of the log info
        when ip address is loaded
        """
        self._countryname =
            GlobalStats.get_country_name(self.ip_addr)

    @property
    def countryname(self) -> str:
        """
        The country name is a property cause
        of needed external informations
        Just read only
        """
        if self._countryname == "" and self.__try_get_country_name:
            self.search_country_name()
            self.__try_get_country_name = False
        return self._countryname
```

```

@property
def datetime_log(self) -> datetime.datetime:
    """
    The date is stored like '2014-12-01 00:22:46'
    in the log file
    This property return an object datetime
    This property is not reusable from another program
    due to the specific format (hardcoded)
    """
    m = LogInfo.re_datetime.match(self.dt)
    return (datetime.datetime.strptime(m.group("dt") + " "
        + m.group("tm"), "%Y-%m-%d %H:%M:%S") if m else "")

@property
def date_log(self) -> str:
    """
    Extract only the date from the date time LogInfo member
    """
    m = LogInfo.re_datetime.match(self.dt)
    return (m.group("dt") if m else "")

def __str__(self):
    return ("Date : {this.dt}; ip_address : {this.ip_addr}; " \
        "Page : {this.page}; Category : {this.category}; " \
        "Pays : {this.countryname}").format(this = self)

def __cmp__(self, obj):
    """
    This object can be compared by his datetime_log property
    """
    if (obj is not None and type(obj) is type(self)):
        return self.datetime_log.__cmp__(obj.datetime_log)
    return -1

def __lt__(self, obj):
    """
    This object can be compared by his datetime_log property
    """
    if (obj is not None and type(obj) is type(self)):
        return self.datetime_log < obj.datetime_log
    return -1

def __gt__(self, obj):
    """
    This object can be compared by his datetime_log property
    """

```


Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
if (obj is not None and type(obj) is type(self)):
    return self.datetime_log > obj.datetime_log
return -1

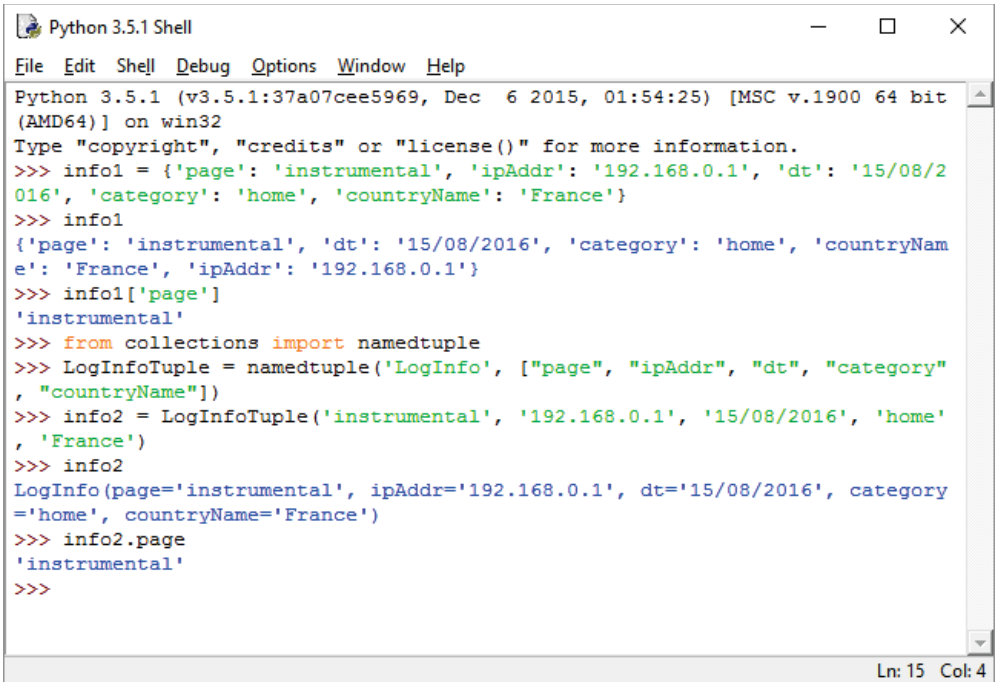
def __eq__(self, obj):
    """
    This object can be compared by his datetime_log property
    """
    if (obj is not None and type(obj) is type(self)):
        return self.datetime_log == obj.datetime_log
    return -1
```

Très petite et n'ayant pas de réelles méthodes de classe métier, cette dernière pourrait être remplacée par un dictionnaire de données, tel que:

```
info1 = {'page': 'instrumental', 'ipAddr': '192.168.0.1', 'dt':
'15/08/2016', 'category': 'home', 'countryName': 'France'}
```

Une autre façon d'écrire ce dictionnaire de données est d'utiliser les Tuple nommés:

```
from collections import namedtuple
LogInfoTuple = namedtuple('LogInfo', ["page", "ipAddr", "dt",
"category", "countryName"])
info2 = LogInfoTuple('instrumental', '192.168.0.1', '15/08/2016',
'home', 'France')
```



```
Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:54:25) [MSC v.1900 64 bit
(AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> info1 = {'page': 'instrumental', 'ipAddr': '192.168.0.1', 'dt': '15/08/2
016', 'category': 'home', 'countryName': 'France'}
>>> info1
{'page': 'instrumental', 'dt': '15/08/2016', 'category': 'home', 'countryNam
e': 'France', 'ipAddr': '192.168.0.1'}
>>> info1['page']
'instrumental'
>>> from collections import namedtuple
>>> LogInfoTuple = namedtuple('LogInfo', ["page", "ipAddr", "dt", "category"
, "countryName"])
>>> info2 = LogInfoTuple('instrumental', '192.168.0.1', '15/08/2016', 'home'
, 'France')
>>> info2
LogInfo(page='instrumental', ipAddr='192.168.0.1', dt='15/08/2016', category
='home', countryName='France')
>>> info2.page
'instrumental'
>>>
```

Illustration 10: Traitement d'un dictionnaire de données

Nous pourrions alors emballer cet objet dans une liste de dictionnaires, mais il faudrait respecter scrupuleusement le nom des clés des différents dictionnaires. Gardons pour l'instant cette philosophie de côté pour l'utiliser en programmation fonctionnelle par la suite.

Après cette classe `LogInfo`, nous allons définir la classe `CountryInfo`. Cette dernière est plus intéressante, car elle implémente une méthode de classe.

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
class CountryInfo(object):
    """
    This object contains the information of a ip address range lent
    by a specific country
    """

    def __init__(self, country_line: list):
        self.ip_start = 0
        self.ip_end = 0
        self.a2 = ""
        self.a3 = ""
        self.name = ""

        # parse the input data
        if len(country_line) > 4:
            self.ip_start, self.ip_end = [int(x) for x
                                           in country_line[0:2]]
            self.a2, self.a3, self.name = country_line[2:5]

    def __str__(self):
        return ("IPStart : {this.ip_start}; IPEnd : {this.ip_end};"\
               " A2 : {this.a2}; A3 : {this.a3}; "\
               "Name : {this.name}").format(this = self)

    def contains_ip(self, ip_addr: str) -> bool:
        """
        Check if the ip_addr parameter is a valide ip v4 adress
        Transform it to a number
        Check if this number is included into this range object
        """
        ip_num = IPHelper.ip_address_to_ip_number(ip_addr)
        if ip_num is None: return False
        return self.contains_ip_num(ip_num)

    def contains_ip_num(self, ip_num: int) -> bool:
        """
        Check if this number is included into this scale object
        """
        return (self.ip_start <= ip_num <= self.ip_end)

    @property
    def countryname(self) -> str:
        return (self.name if (self.name != "") else self.a2)
```

La méthode (containtIP) est assez intéressante, car elle permet de dire si le pays utilisé contient l'adresse IP passée en paramètres.

Nottions l'introduction de la propriété 'CountryName' renvoyant le nom du pays de l'objet en cours.

Passons ensuite à la classe statique GlobalStats.

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
class GlobalStats(object):
    """
    This class is used like a singleton (all is static)
    Contains the data from Countries Informations,
    IP Address, and also all logs to analyze
    """

    iplist = {}
    countryinfos = []
    pages = {}
    categories = {}
    countrynamecount = {}
    countrynamedetails = {}
    visites_dates = {}
    logs = []

    @staticmethod
    def calc_stats(loginfos: list):
        """
        Get a specific array, iter on it and create statistics
        Splitted and Used for ThreadPoolExecutor
        """
        for loginfo in loginfos: GlobalStats.set_stat(loginfo)

    @staticmethod
    def set_stat(loginfo: LogInfo):
        """
        This function calcul a statistic for the LogInfo object
        """
        GlobalStats.pages[loginfo.page] =
            GlobalStats.pages.get(loginfo.page, 0) + 1
        GlobalStats.categories[loginfo.category] =
            GlobalStats.categories.get(loginfo.category, 0) + 1
        GlobalStats.visites_dates[loginfo.date_log] =
            GlobalStats.visites_dates.get(loginfo.date_log, 0) + 1

        if loginfo.countryname != "":
            GlobalStats.countrynamecount[loginfo.countryname] =
                GlobalStats.countrynamecount.get(
                    loginfo.countryname, 0) + 1

            if not(loginfo.countryname in
                    GlobalStats.countrynamedetails):
                GlobalStats.countrynamedetails[loginfo.countryname]
                    = {}
```

```

        GlobalStats.countrynamedetails[logininfo.countryname]
            [logininfo.ip_addr] = \

GlobalStats.countrynamedetails[logininfo.countryname].get(
            logininfo.ip_addr, 0) + 1

@staticmethod
def start_analysis():
    """
    Clear all statistics arrays on GlobalStats
    and start analysis
    """
    # clear all analysis arrays
    GlobalStats.pages.clear()
    GlobalStats.categories.clear()
    GlobalStats.countrynamecount.clear()
    GlobalStats.countrynamedetails.clear()
    # sort the logs by date
    GlobalStats.logs.sort()
    # start the analysis
    GlobalStats.calc_stats(GlobalStats.logs)

@staticmethod
def get_country_name(ip_addr: str) -> str:
    """
    Return the name of the country linked to the ip_address
    """
    ip_num = IPHelper.ip_address_to_ip_number(ip_addr)

    if (not(ip_num in GlobalStats.iplist)):
        # get the first or default country name with specific IP
        # do not use the fonctionnal programming
        # for this algorithme
        GlobalStats.iplist[ip_num] = "__ Not Registered __"
        for countryLineTmp in GlobalStats.countryinfos:
            if (countryLineTmp.contains_ip(ip_num)):
                GlobalStats.iplist[ip_num] =
                    countryLineTmp.countryname
                break

    return GlobalStats.iplist[ip_num]

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
@staticmethod
def load_informations(filename: str, destarray: list,
                      delegate_class: type, charDelimiter: str = ';'):
    """
    This function take a file name and parse
    for each line the document
    Transforme in a specific objet
    and add into an destination array
    filename      : path and name of a csv file to import
    destarray     : the array which be populated
    delegate_class : the class to instanciate each line
    charDelimiter : the delimiter from csv file
    """
    with open(filename, "r") as fic:
        try:
            reader_object = csv.reader(fic,
                                       delimiter=charDelimiter, quotechar='"')
            destarray += [delegate_class(data) for data
                          in reader_object if len("".join(data)) > 0]
        except:
            print("Problem on reading csv.reader " \
                  "{0}".format(filename))
```

Toutes ses méthodes internes sont statiques, car c'est un regroupement de fonctions procédurales dans un conteneur.

Poursuivons le code par l'utilisation d'une classe d'aide au travail sur les adresses Ipv4.

```

class IPHelper(object):
    """
    The IPHelper class is about the IP treatment:
    - concert a IP Number to IP patern and reverse
    - check a IP String
    """

    # ip v4 address patern
    re_ip = re.compile(r'^\d{1,3}(\.\d{1,3}){3}$')

    @staticmethod
    def ip_address_to_ip_number(dotted: str) -> int:
        """
        This function ip_address_to_ip_number
        get an IPv4 address (e.g. 192.168.0.1)
        And translate it as a number
        """
        if not IPHelper.is_valide_ip_address(dotted): return None

        dotted = re.split( '\.', dotted)
        return sum([int(dotted[x]) * pow(0x100, (3-x))
                    for x in range(0,4,1)])

    @staticmethod
    def ip_number_to_ip_address(number: int) -> str:
        """
        This function ip_number_to_ip_address get a number
        And translate it as an IPv4 address (e.g. 192.168.0.1)
        a = (number / 0x1000000) % 0x100
        b = (number / 0x10000) % 0x100
        c = (number / 0x100) % 0x100
        d = (number / 0x1) % 0x100
        """
        return ".".join([(str((number // pow(0x100, (3-x)))) % 0x100)
                        for x in range(0,4,1)])

    @staticmethod
    def is_valide_ip_address(ip_addr: str) -> bool:
        """
        Check if the string can match with a IP patern
        """
        reMatch = IPHelper.re_ip.match(ip_addr)
        return reMatch is not None and [0 <= int(x) < 256 for x
                                         in re.split('\.', reMatch.group(0))].count(True) == 4

```

Puis, définissons deux fonctions d'aide à la récupération de fichiers; une pour les

Analysez (vos besoins), Etudiez, Scriptez L'art d'utiliser du scripting pour un projet

informations sur les ip des différents pays, l'autre pour la liste des fichiers de log.

```
def HelperFunctions():
    """
    HelperFunctions return a tuple with functions witch can be use
    on the main program as a macro function
    """

    def get_last_recent_ipcountries_list() -> str:
        """
        Get file about defined of ip country (more recent)
        """
        return max([ipf for ipf in listdir(abspath(".")) if
                     ipf.startswith("ip-to-country") and ipf.endswith(".csv")])

    def get_array_log_files_names(filter_name: str,
                                   directory_logs: str) -> list:
        """
        Get array of ip to analyse
        Get all file logs in directory
        Get each log file entry
        """
        logs_files_names = [join(directory_logs, filelog)
                             for filelog in listdir(directory_logs)
                             if isfile(join(directory_logs, filelog))
                             and filelog.endswith(".log")] #\

        # filter the date if not 'all'
        if filter_name != "all":
            logs_files_names = list(filter(lambda dt: filter_name in
                                             dt, logs_files_names))

        return logs_files_names

    return (get_last_recent_ipcountries_list,
            get_array_log_files_names)
```

Passons maintenant au programme principal permettant de récupérer les fichiers log, charger le fichier d'informations sur les adresses IP réservées par pays, et du lancement des statistiques.

```

def main():
    # -----
    # MAIN

    # management on the bash starting
    # "/?"; "--help"; "-h" or "yyyyMM" of log statistics
    date_logs = sys.argv[1] if len(sys.argv) > 1 else "all"
    # (count(argv) > 1)?argv[1]:"all"
    if date_logs in ["--help", "/?", "-h"]:
        print("Aide sur StatLog")
        print("=====")
        print("./statLog3.py [date yyyy[MM]] [directory_logs] " \
              "[-nographic]\n")
        exit(0)
    elif not( not(re.fullmatch("(\\d){4}|(\\d){6}", date_logs)
               is None) or date_logs == "all" ):
        strTime = strftime("%Y%m", gmtime())
        print("Erreur du format de la date : yyyyMM " \
              "(ex: {0})\n".format(strTime))
        exit(-1)

    # 2nd argument is the path of directory logs "default == ./logs"
    directory_logs = sys.argv[2] if len(sys.argv) > 2 and
        not str(sys.argv[2]).startswith("-") else abspath("./logs")
    # (count(argv) > 2)?argv[2]:"./logs"
    if(not(isdir(directory_logs))):
        print("Erreur d'ouverture du repertoire "
              + directory_logs + " !\n")
        exit(-1)

    # get information about graphical printing part
    show_graphical_part = not ("--nographic" in sys.argv)

    startTime = time()

    print("Statistiques de connection" \
          "\n=====\\n")

    # get many functions for the files treatment
    last_recent_ip_file, array_log_files_names = HelperFunctions()

    # import the countries information
    GlobalStats.load_informations(last_recent_ip_file(),
        GlobalStats.countryinfos, CountryInfo, charDelimiter = ',')
    timeLoadCountry = time()

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
# get all log files names to analyze
logs_files_names = array_log_files_names(date_logs,
                                           directory_logs)

# exit if not log files with filter name
if len(logs_files_names) == 0:
    print("File(s) not found %s/logConnexion%s.log"
          % (directory_logs, date_logs))
    exit(-1)

# import the logs rows from on or many files
for entryName in logs_files_names:
    GlobalStats.load_informations(entryName, GlobalStats.logs,
                                  LogInfo, charDelimiter = ';')

timeLoadFile = time()

# country name: get the name of country foreach row
# in the log files
for tmpinfo in GlobalStats.logs: tmpinfo.search_country_name()

timeCountryName = time()

# start analysis
GlobalStats.start_analysis()

timeAnalysis = time()

# count the times for each pages
countNbVisitor = sum([len(tmpArr) for tmpArr
                      in GlobalStats.countrynamedetails])

# Print statistics
print("Nombre de visites par jour :")
pprint(GlobalStats.visites_dates)
print("Moyenne de visites par jour : %.2f / " \
      "Top Connections sur 1 jour : %d" \
      % (mean([v for v in GlobalStats.visites_dates.values()]),
         max(GlobalStats.visites_dates.values()))))
print("Detail des adresses par pays :")
pprint(GlobalStats.countrynamedetails)
print("Nombre de visiteurs : %d" % (countNbVisitor))
print("Resume des adresses par pays :")
pprint(GlobalStats.countrynamecount)
print("Total : %d\n" % (sum([x for x
                             in GlobalStats.countrynamecount.values()])))
```

```

print("Nombre de visites par page :")
pprint(GlobalStats.pages)
print("Total : %d pages\n" % (sum([y for y
                                in GlobalStats.pages.values()])))
print("Nombre de visites par categorie :")
pprint(GlobalStats.categories)
print("Total : %d pages / %d categories\n" % (sum([z for z
                                in GlobalStats.categories.values()]),
                                len(GlobalStats.categories)))
print("Fin des statistiques\n=====\n")

# end of script
endTime = time()

print("Temps total du script      : %10.5f"
      % (endTime - startTime))
print("Temps chargement pays      : %10.5f"
      % (timeLoadCountry - startTime))
print("Temps chargement fichier   : %10.5f"
      % (timeLoadFile - timeLoadCountry))
print("Temps recuperation pays    : %10.5f"
      % (timeCountryName - timeLoadFile))
print("Temps calcul des stats     : %10.5f"
      % (timeAnalysis - timeCountryName))
print("Temps stats affichage      : %10.5f"
      % (endTime - timeAnalysis))

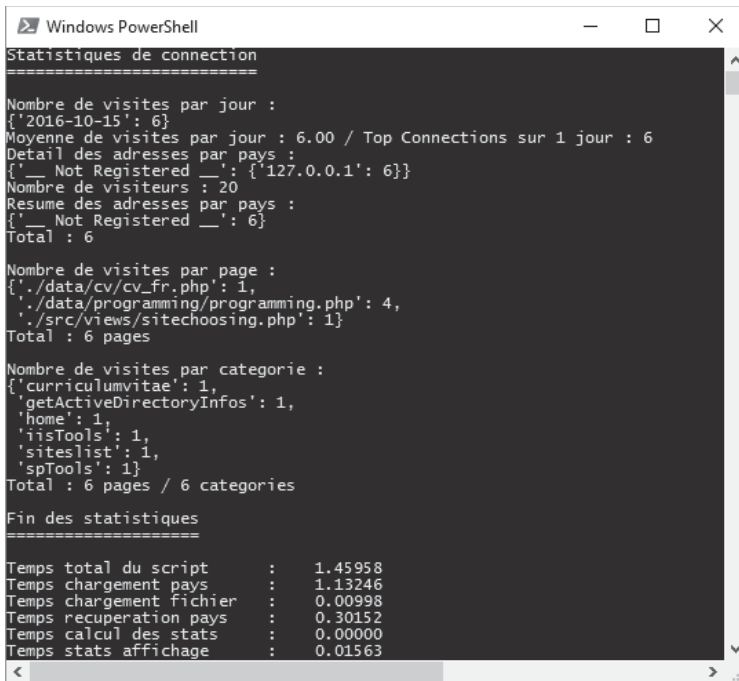
if show_graphical_part: view_graphical_stats()

if __name__ == '__main__':
    main()

```

Analysez (vos besoins), Etudiez, Scriptez L'art d'utiliser du scripting pour un projet

Exemple de résultat du programme sur un fichier 'dummy'.



```
Windows PowerShell

Statistiques de connection
=====

Nombre de visites par jour :
{'2016-10-15': 6}
Moyenne de visites par jour : 6.00 / Top Connections sur 1 jour : 6
Detail des adresses par pays :
{'__ Not Registered __': {'127.0.0.1': 6}}
Nombre de visiteurs : 20
Resume des adresses par pays :
{'__ Not Registered __': 6}
Total : 6

Nombre de visites par page :
{'./data/cv/cv_fr.php': 1,
 './data/programming/programming.php': 4,
 './src/views/sitechoosing.php': 1}
Total : 6 pages

Nombre de visites par categorie :
{'curriculumvitae': 1,
 'getActiveDirectoryInfos': 1,
 'home': 1,
 'iisTools': 1,
 'siteslist': 1,
 'spTools': 1}
Total : 6 pages / 6 categories

Fin des statistiques
=====

Temps total du script      : 1.45958
Temps chargement pays     : 1.13246
Temps chargement fichier  : 0.00998
Temps recuperation pays   : 0.30152
Temps calcul des stats    : 0.00000
Temps stats affichage     : 0.01563
```

Illustration 11: Exemple de Calcul de statistiques

Interaction entre le Scripting et une interface graphique

Face au précédent programme, en mode texte et ligne de commande, nous pouvons simplement demander à Python un petit coup de main graphique, pour afficher un graphe avec le nombre de connexions par jour.

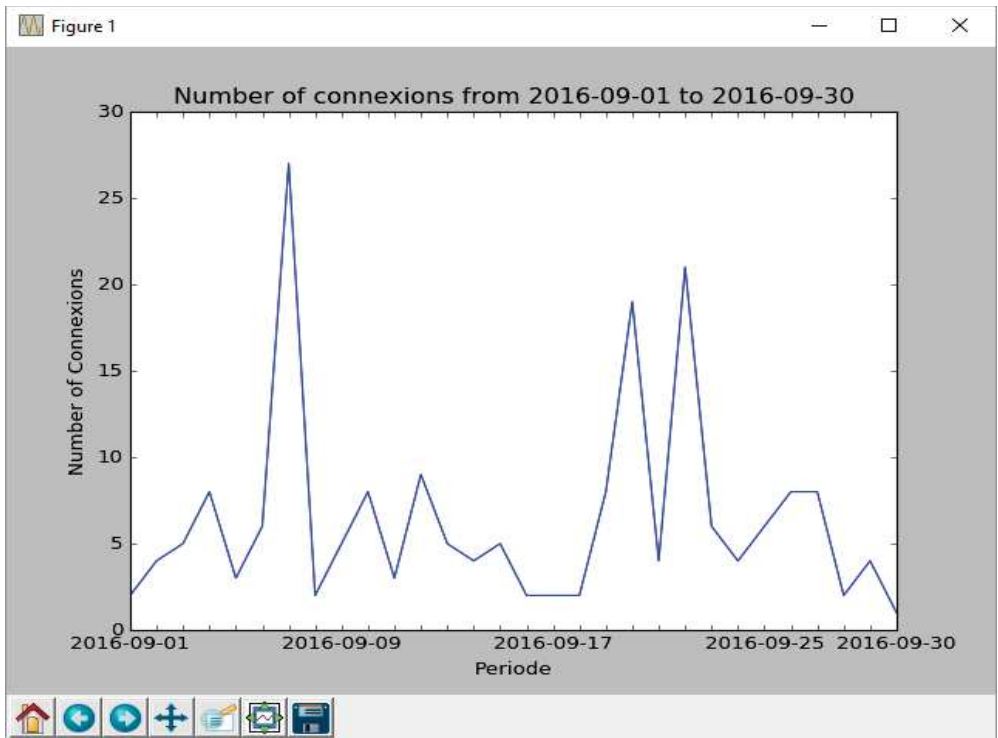


Illustration 12: Interface graphique intégrée à Python (TK)

Python intègre dans ses librairies par défaut le langage graphique Ticket (TK). Ce dernier est très pratique et est utilisé pour le module MathPlotLib, module au combien merveilleux!

Analysez (vos besoins), Etudiez, Scriptez L'art d'utiliser du scripting pour un projet

Ajoutons simplement une petite fonction au programme précédent permettant de récupérer les connexions en relation avec le temps.

```
def view_graphical_stats():
    """
    Use the graphical Python TK component to show
    the curve of connections
    """

    # graphic part
    nb_printing_dt = 4
    x_axis = [k for k in GlobalStats.visites_dates.keys()]
    x_axis.sort()
    # create labels for the X axis
    # insert empty string due to so much of printed data
    ratio_printing = round(len(x_axis) / nb_printing_dt)
    x_axis_printed = [p if i % ratio_printing == 0 else ""
                      for i,p in enumerate(x_axis)]
    # write the last date
    x_axis_printed[-1] = x_axis[-1]
    # extract the Y axis
    y_axis = [int(GlobalStats.visites_dates[v]) for v in x_axis]

    # create the graphical figure
    fig = plt.figure()
    # axis for print the date as string
    x_ticks = range(len(x_axis))
    plt.xticks(x_ticks, x_axis)
    locs, labels = plt.xticks()
    plt.xticks(locs, x_axis_printed)
    plt.setp(labels, rotation=0)
    # plot the curve
    plt.plot(x_ticks, y_axis)
    # print labels and title
    plt.xlabel('Periode')
    plt.ylabel('Number of Connexions')
    plt.title("Number of connexions from %s to %s"
              % (min(x_axis), max(x_axis)))

    # show
    plt.show()
    # dispose manually, cause of memory fails
    plt.close()
```

Rien qu'avec ce petit bout de code, nous avons une entrée en matière pour un affichage graphique élaboré. Donc, simplicité, rapidité et efficacité.

Utiliser GTK+ et Python

Comme nous l'avons vu précédemment, si l'interface graphique est suffisamment bien séparée du programme, il est possible d'utiliser un langage graphique identique pour deux programmes distincts, tel que GTK+, par exemple.

Avec PHP, il est extrêmement simple de coupler GTK+. Avec Python, cela est encore plus aisé.

Les pré-requis pour l'installation:

- Sous Windows: installer le package PyGtk
- Sous Linux: la commande "apt-get install python-gtk python-glade2" vous permettra d'installer les deux modules liés à GTK+.

NB: un installateur dédié à python est maintenant disponible sous Linux. Ce dernier s'appelle "pip" et s'installe tel que "apt-get install pip". Ensuite, si l'on désire installer un packet particulier (par exemple matplotlib), il suffit de taper "pip install matplotlib".

Exemple de programme: Car Cost Simulator

Ce programme est un simulateur de dépenses liées à une ou plusieurs voitures. Créer au départ pour comparer le coût d'utilisation de deux voitures afin de faire un choix financier d'achat de véhicule.

Cahier des charges :

Créer un logiciel permettant de calculer le budget de la vie d'une voiture.

Possibilité d'ajouter des composants au logiciel après développement (plug-in).

Créer une interface générique capable d'afficher des objets inconnus au moment du développement.

Besoin multi plate-forme (Unix, Windows).

Couper court aux idées reçues :

Par exemple, simuler le coût d'un véhicule. Vous allez dire « ok, je vais acheter une nouvelle voiture. Je souhaite un modèle diesel, car je roule environ 1000 kilomètres par an. ».

Alors, la personne moyenne décide d'acheter un véhicule diesel, 2000 euros plus cher que le même modèle en essence, car elle estime qu'avec le prix plus bas du diesel, le modèle sera rentabilisé dans la durée.

Avec ce simulateur, nous pourrions voir, de manière factuelle, que ce que l'on pensait être rentable ne l'est pas forcément. On pourra comprendre qu'il faut parfois des dizaines d'années pour rentabiliser le surcoût d'achat d'un diesel alors que le véhicule sera gardé en moyenne six années.

Analyse

Avec le cahier des charges, nous comprenons que le système de réflexion par le code sera une part très importante dans ce projet, bien plus que le codage pur.

Java ne me convenait pas trop au niveau de la réflexion de code. Scala aurait été bien, mais je ne savais pas comment réaliser l'analyse de la situation.

J'ai essayé de développer l'application en DotNet, mon langage quotidien chez mes clients, mais après plusieurs jours de POC, j'ai décidé de passer à un autre langage, le Python.

J'ai fait ce choix pour les raisons de tester de très petites parties de code. Par exemple, créer une classe « Car » et la tester de suite, comme un logiciel à part. Réactivité directe, car je développais en même temps que j'exécutais le code (possible en 2016 avec Visual Studio, mais pas en 2012). Le codage à la volée.

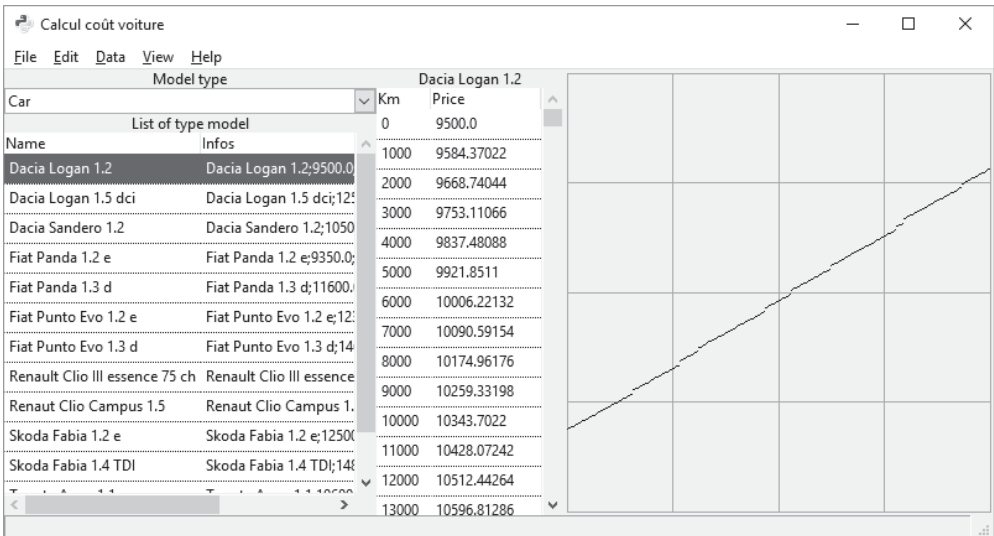


Illustration 13: Page principale de CarCostSimulator

Dans un premier temps, analysons l'architecture du programme:

D'un côté, nous avons nos bibliothèques python compilées en python pure ('.pyc').

Ces bibliothèques représentent le comportement du simulateur.

De l'autre, nous avons la description de l'interface graphique dans un (ou plusieurs) fichier XML.

Entre les deux, nous utilisons l'API PyGtk, avec une légère surcouche de python pour adapter l'interface à notre comportement logiciel.

Côté métier, nous avons un module nommé "Modelizing", contenant les classes "Car", "Fuel", "Driver", "DriverType".

La classe "Car" définit la voiture; la classe "Fuel" définit le type de carburant (essence, diesel, gpl, ...); la classe "Driver" définit le conducteur; la classe "DriverType" définit la manière de conduire. Cela influe sur la consommation donc sur le résultat financier final et se traduit par un coefficient de consommation lié à "Driver" et à "Fuel". Chacune de ces classes héritera de deux classes "ListManaged" et "XmlExp". L'une gère les listes d'objets et l'autre gère l'export en Xml pour sauvegarde du travail.

Analysons à présent la partie du module nommée "ObjectModelized". Cette partie contient la base des classes utilisées pour les classes métier.

```
# Module ObjectModelized.py

import pylab as pl
import inspect
from types import *
from copy import copy, deepcopy
import logging

logger = logging.getLogger("CarCostSimulator")

#To transform in interface (with Python update 3)
#Now, compatible to Python 2.6
class XmlExp(object):

    def __init__(self):
        pass

    def XmlExport(self, indent = 0, objID = -1):
        if objID == -1:
            objID = id(self)
        data = "<" + self.__class__.__name__ + " id=\"\"
            + str(objID).rjust(2, '0') + \"\" name=\"\"
            + str(self) + \">\"
        xmltxt = data.rjust(len(data) + indent, ' ') + \"\\n\"
        tmpMembers = inspect.getmembers(self.__class__)

        for mbr in tmpMembers:
            propertyType, propertyName = (type(mbr[1]),str(mbr[0]))
            propertyObj = self.__getattr__(propertyName)

            # if the property is not a list of contents
            if propertyType.__name__ == "property"
            and type(propertyObj) is not ListType:
                data = "<" + propertyName + ">" + str(propertyObj)
                + "</" + propertyName + ">"
                xmltxt += data.rjust(len(data)
                    + (indent + 2), ' ') + \"\\n\"

            # if it's a content list of objects
            elif propertyType.__name__ == "property":
                tmpData = ""
                idxSubObj = 1
                for subobj in propertyObj:
                    if hasattr(subobj, 'XmlExport'):
```

Interaction entre le Scripting et une interface graphique

```

        xmltxt += subobj.XmlExport(indent + 2,
                                   idxSubObj) + "\n"
        idxSubObj += 1

    #if tmpData != "":
        # parent ancrer
        #data = "<" + propertyName + ">"
        #xmltxt += data.rjust(len(data)
        #        + (indent + 2), ' ') + "\n"
        # child ancrer
        #xmltxt += tmpData
        # end of parent ancrer
        #data = "</" + propertyName + ">"
        #xmltxt += data.rjust(len(data)
        #        + (indent + 2), ' ') + "\n"

    data = "</" + self.__class__.__name__ + ">"
    xmltxt += data.rjust(len(data) + indent, ' ')
    return xmltxt

```

```

class ListManaged(object):

    def __init__(self, name = "") :
        self._name = name
        if not hasattr(self.__class__, 'arrayObj'):
            setattr(self.__class__, 'arrayObj', {})

    def __str__(self):
        return self._name

    @property
    def name(self):
        return self._name
    @name.setter
    def name(self, value):
        if type(self.__class__.arrayObj) is dict
            and self in self.__class__.arrayObj.values():
            # delete entry
            self.__class__.removeObj(self)
            # rename object
            self._name = value
            # adding with new key
            self.__class__.appendObj(self)
        else:
            self._name = value

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
def __cmp__(self, itemCmp):
    return cmp(str(self).lower(), str(itemCmp).lower())

@classmethod
def appendObj(cls, newObj):
    if not hasattr(cls, "arrayObj"):
        return False

    cls.arrayObj[str(newObj)] = newObj
    logger.debug("Append " + str(newObj) + " of "
                 + cls.__name__)

@classmethod
def updateObj(cls, oldObj, newObj):
    if not hasattr(cls, "arrayObj"):
        return False
    if str(oldObj) == str(newObj):
        cls.arrayObj[str(oldObj)] = newObj
    else:
        cls.removeObj(oldObj)
        cls.appendObj(newObj)

@classmethod
def removeObj(cls, objToRemove):
    if not hasattr(cls, "arrayObj"):
        return False

    objname = str(objToRemove)
    if cls.arrayObj.has_key(objname):
        logger.debug("Remove " + str(cls.arrayObj[objname]))
        del cls.arrayObj[objname]
        return True
    return False

@classmethod
def clearObj(cls):
    if not hasattr(cls, "arrayObj"):
        return False
    elif len(cls.arrayObj) == 0:
        return False

    cls.arrayObj.clear()
    return True
```

Passons à la classe "Car", faisant aussi référence à deux classes internes "CostKm" et "Wearpart".

```
# Module CarModelized.py

import pylab as pl
import inspect
from types import *
from copy import copy, deepcopy
from ObjectModelized import *
from FuelModelized import *
from DriverModelized import *
import sys
import logging

class CostKm(object):

    def __init__(self, kilometers = 0.0, price = 0.0):
        self.km = kilometers
        self.price = price

    def __cmp__(self, itemCmp):
        return cmp(self.km, itemCmp.km)

    @property
    def km(self):
        return self._km
    @km.setter
    def km(self, value):
        self._km = value

    @property
    def price(self):
        return self._price
    @price.setter
    def price(self, value):
        self._price = float(value)
```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
class Wearpart(XmlExp):
    """
    Class permettant de simuler toute piece d'usure
    (couroie, filtres, ...)
    """

    def __init__(self, nomPiece, periodicity = 0.0, price = 0.0):
        self.name = nomPiece
        self.periodicity = periodicity
        self.price = price

    def __str__(self):
        return self._name

    def getInfos(self):
        return (str(self) + ";" + str(self.price) + ";"
                + str(self.periodicity))

    def __copy__(self, memo=None):
        myClone = Wearpart(self.name, self.periodicity, self.price)
        return myClone

    def __deepcopy__(self, memo=None):
        myClone = Wearpart(self.name, self.periodicity, self.price)
        return myClone

    @property
    def name(self):
        return self._name
    @name.setter
    def name(self, value):
        self._name = value

    @property
    def periodicity(self):
        return (self._periodicity)
    @periodicity.setter
    def periodicity(self, value):
        self._periodicity = float(value)

    @property
    def price(self):
        return self._price
    @price.setter
    def price(self, value):
        self._price = float(value)
```

```

class Car(ListManaged, XmlExp):
    """
    Class Car simulant les principales caracteristiques de la Car
    en fonction du temps, de l'entretien et du cout general
    """

    # variables de classe
    maxkilometers = 301000
    espacementKm = 1000
    nbKmParAnnee = 47000
    maxSizeLengthName = 1
    ids = {}

    def __init__(self, carName = "newCar"):
        ListManaged.__init__(self, carName)
        self._price = 0.0
        self._costs = []
        self._wearparts = []
        self._fuel = None
        self._driver = None
        self._drivertype = None
        self._tanksized = 0.0
        self._insuranceprice = 0.0
        self._consumption = 0.0
        Car.ids[self._name] = -1

    def getInfos(self):
        infoFuel = ""
        if type(self.fuel) is Fuel:
            infoFuel = self.fuel.getInfos()
        else:
            infoFuel = str(self.fuel)
        return (str(self) + ";" + str(self.price) + ";" +
str(self.consumption) + ";" + infoFuel)

    def __copy__(self):
        myClone = Car()
        myClone.name = self.name
        myClone.price = self.price
        myClone.Wearparts = copy(self.Wearparts)
        myClone.fuel = str(self.fuel)
        myClone.driver = str(self.driver)
        myClone.tanksized = self.tanksized
        myClone.insuranceprice = self.insuranceprice
        myClone.consumption = self.consumption
        return myClone

```


Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
def __deepcopy__(self, memo=None):
    myClone = Car()
    myClone.name = self.name
    myClone.price = self.price
    myClone.Wearparts = deepcopy(self.Wearparts)
    myClone.fuel = str(self.fuel)
    myClone.driver = str(self.driver)
    myClone.tanksize = self.tanksize
    myClone.insuranceprice = self.insuranceprice
    myClone.consumption = self.consumption

    return myClone

@property
def name(self):
    return self._name
@name.setter
def name(self, value):
    self._name = value
    Car.ids[self.name] = -1
    if Car.maxSizeLengthName < len(self._name):
        Car.maxSizeLengthName = len(self._name)

# prix d'achat de la voiture
@property
def price(self):
    return self._price
@price.setter
def price(self, value):
    self._price = float(value)

# fuel type
@property
def fuel(self):
    if self._fuel is None:
        return Fuel()
    elif type(self._fuel) is not Fuel:
        self.fuel = self._fuel
    return self._fuel
@fuel.setter
def fuel(self, value):
    try:
        if Fuel.arrayObj.has_key(str(value)):
            self._fuel = Fuel.arrayObj[str(value)]
        else:
            self._fuel = value
```

```

    except:
        self._fuel = value
        logger.warning("Car.fuel setter : "
                        + str(sys.exc_info()))
    Car.ids[self.name] = -1
    @fuel.deleter
    def fuel(self):
        self._fuel = None

    # driver
    @property
    def driver(self):
        if self._driver is None:
            return Driver()
        elif type(self._driver) is not Driver:
            self.driver = self._driver
        return self._driver
    @driver.setter
    def driver(self, value):
        try:
            if hasattr(Driver, 'arrayObj')
               and Driver.arrayObj.has_key(str(value)):
                self._driver = Driver.arrayObj[str(value)]
            else:
                self._driver = value
                Car.ids[self.name] = -1
        except:
            self._driver = value
            logger.warning("Car.driver setter : "
                            + str(sys.exc_info()))
    @driver.deleter
    def driver(self):
        self._driver = None

    # car consumption
    @property
    def consumption(self):
        return self._consumption
    @consumption.setter
    def consumption(self, value):
        self._consumption = float(value)

    # taille du reservoir
    @property
    def tanksize(self):
        return self._tanksize

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
@tanksize.setter
def tanksize(self, value):
    self._tanksize = float(value)

# assurance par an
@property
def insuranceprice(self):
    return self._insuranceprice
@insuranceprice.setter
def insuranceprice(self, value):
    self._insuranceprice = float(value)

# pieces d'usures
@property
def Wearparts(self):
    return self._wearparts
@Wearparts.setter
def Wearparts(self, value):
    del self.Wearparts
    self._wearparts = value
@Wearparts.deleter
def Wearparts(self):
    del self._wearparts[:]

# Tableau des couts
@property
def Costs(self):
    tmpId = id(self) + id(self.fuel) + id(self.driver)
    if Car.ids[self.name] != tmpId:
        del self._costs[:]
        Car.ids[self.name] = tmpId
    if len(self._costs) == 0:
        self._costs = self.calculCouts()
    #extract only the cost part of CostKm object
    try:
        return [float(x.price) for x in self._costs]
    except:
        Car.ids[self.name] = -1
        return []

def calculCouts(self):
    tmpCalc = []
    tmpCalc.append(CostKm(0.0,self.price))
    drivingcoef = 1.0
    # get driving coeeficient if it's possible
    if type(self.driver) is Driver:
```

```

        if type(self.driver.drivertype) is Drivertype:
            drivingcoef =
                self.driver.drivertype.drivingcoefficient
        if not type(tmpCalc[-1]) is CostKm:
            return []
        while tmpCalc[-1].km < Car.maxkilometers :
            previousCostKm = tmpCalc[-1]
            actualCostKm = CostKm(previousCostKm.km
                                   + Car.espacementKm, previousCostKm.price)
            # cout par kilometres
            if type(self.fuel) is Fuel:
                actualCostKm.price += Car.espacementKm
                                   * self.consumption / 100.0
                                   * self.fuel.price * drivingcoef
            # calcul des prix des pieces d'usure
            for pceUsure in self.Wearparts :
                if pceUsure.periodicity != 0 and (actualCostKm.km
                                                    % pceUsure.periodicity) == 0.0:
                    actualCostKm.price += pceUsure.price
                                           * drivingcoef
            # cout par annee assurance
            if ((actualCostKm.km + self.nbKmParAnnee)
                % self.nbKmParAnnee) == 0:
                actualCostKm.price += self.insuranceprice
            # ajout cout global
            tmpCalc.append(actualCostKm)
        return tmpCalc

def refreshCalc(self):
    Car.ids[self.name] = -1
    return

def __lt__(self, other):
    if type(self) == type(other):
        return self.Costs[-1] < other.Costs[-1]
    return False

def __le__(self, other):
    if type(self) == type(other):
        return self == other or self < other
    return False

def __gt__(self, other):
    if type(self) == type(other):
        return self.Costs[-1] > other.Costs[-1]
    return False

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
def __ge__(self, other):
    if type(self) == type(other):
        return self == other or self > other
    return False

def __eq__(self, other):
    if type(self) == type(other):
        return self.Costs[-1] == other.Costs[-1]
    return False

def __ne__(self, other):
    if type(self) == type(other):
        return self.Costs[-1] != other.Costs[-1]
    return False

def kmCmp(self, other):
    if type(self) != type(other):
        return False

    tmpCostsV1 = self.Costs
    tmpCostsV2 = other.Costs
    cmpVoit = cmp(self.price, other.price)
    idx = 1
    lenTabKm = Car.maxkilometers / Car.espacementKm
    if lenTabKm > len(tmpCostsV1) or lenTabKm > len(tmpCostsV2):
        return cmpVoit
    while idx < lenTabKm:
        cmpVoitTmp = cmp(tmpCostsV1[idx], tmpCostsV2[idx])
        if cmpVoit != cmpVoitTmp:
            return cmpVoitTmp * Car.espacementKm * idx
        idx += 1
    return cmpVoit

@staticmethod
def calcTabDiffAmortissements():
    if not hasattr(Car, 'arrayObj'):
        return [], {}

    if len(Car.arrayObj) <= 1:
        return [], {}

    tabAmortissement = []
    sizeofMaxNameTabCar = Car.maxSizeLengthName

    matriceCars = {}
```

Interaction entre le Scripting et une interface graphique

```

for voit1 in Car.arrayObj.values():
    tabHorizon = {}
    for voit2 in Car.arrayObj.values():
        strCmpKm = ""
        tmpKm = voit1.kmCmp(voit2)
        if tmpKm > 0:
            strCmpKm = ">"
        elif tmpKm < 0:
            strCmpKm = "<"
        elif tmpKm == 0:
            strCmpKm = "="
        else:
            strCmpKm = "!"
        tabHorizon[str(voit2)] = tmpKm

        strTmpKm = str(abs(tmpKm))
        if strTmpKm == "1": strTmpKm = "/"
        lineAmortissement = adjustCaract(voit1) + " "
            + strCmpKm + " " + adjustCaract(voit2)
            + " ==> " + strTmpKm.rjust(6)
        tabAmortissement.append(lineAmortissement)
    matriceCars[str(voit1)] = tabHorizon
return tabAmortissement, matriceCars

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
# globals lambda functions
adjustCaract = lambda objTemp:
str(objTemp).ljust(Car.maxSizeLengthName)

def sauvComparaisonCSV(strFileName):
    fic = open(strFileName, "w")
    fic.write("Kilometres;" + ";" + ".join(Car.arrayObj.keys()) + "\n")

    idx = 0
    lentab = Car.maxkilometers / Car.espacementKm
    while idx < lentab:
        fic.write(str(idx * Car.espacementKm) + ";")
        for voit in Car.arrayObj.values():
            fic.write(str(voit.Costs[idx]) + ";")
        fic.write("\n")
        idx +=1
    fic.close()

def printCarsCompaired():
    if not hasattr(Car, 'arrayObj'):
        return
    print("Liste des differentes voitures comparees")
    getCarInfos = lambda objCar: objCar.getInfos()
    carList = map(getCarInfos, Car.arrayObj.values())
    print("\n".join(carList))
    return

def showCurves():
    if not hasattr(Car, 'arrayObj') or len(Car.arrayObj) == 0:
        return False

    # creating absices line
    x = list(range(0, Car.maxkilometers + Car.espacementKm,
Car.espacementKm))
    plotCurves = lambda objCar: pl.plot(x, objCar.Costs)
    map(plotCurves, Car.arrayObj.values())
    pl.xlabel = "Kilometers"
    pl.ylabel = "Cost"
    pl.show()
    pl.close()
    return True

def sauvComparaisonText(strFileName = ""):
    tabAmort, matriceCars = Car.calcTabDiffAmortissements()
    sizeofMaxNameTabCar = Car.maxSizeLengthName
```

```

fic = open(strFileName, "w")
#header
fic.write("Matrice de comparaison\n=====\n")
strLigne = "| " + adjustCaract(" ") + " |"
for voit1 in Car.arrayObj.keys():
    strLigne += "| " + adjustCaract(voit1) + " |"
fic.write(strLigne + "\n")

#content
for voit1 in Car.arrayObj.keys():
    strLigne = "| " + adjustCaract(voit1) + " |"
    for voit2 in Car.arrayObj.keys():
        strLigne += "| "
            + adjustCaract((matriceCars[voit1][voit2]) + " |"
    fic.write(strLigne + "\n")

#header 2
fic.write("\n\nComparaison Voitures\n=====\n")
#content 2
fic.write("\n".join(tabAmort))
# close fic
fic.close()

```


Analysez (vos besoins), Etudiez, Scriptez L'art d'utiliser du scripting pour un projet

Ne pas oublier les tests unitaires (bien plus intéressants dans ce langage qu'avec C#)!

```
def main():
    fuelD = Fuel("diesel", 1.39)
    print fuelD.getInfos()
    Fuel.arrayObj[str(fuelD)] = fuelD
    skoda = Car("Skoda Fabia 1.4 TDI")
    skoda.price = 14800.0
    skoda.fuel = "diesel"
    skoda.consumption = 4.7
    skoda.tanksize = 43.0
    skoda.insuranceprice = 534.0
    skoda.Wearparts.append(Wearpart("Entretien", 15000, 120.0))
    skoda.Wearparts.append(Wearpart("Courroie", 90000, 750.0))
    print skoda.getInfos()
    print skoda.XmlExport()
    Car.appendObj(skoda)
    #Car.showCurves()

#do for unit tests
if __name__ == '__main__':
    main()
```

Python propose aussi une bibliothèque complète dédiée aux tests unitaires, vous pourrez vous renseigner, elle est vraiment très simple et bien conçue.

Passons maintenant à la gestion du conducteur.

Deux classes présentées ci-dessous, permettant de définir le conducteur. Le nombre de kilomètres par an, le nombre total de kilomètres sur la prochaine voiture.

```
# Module DriverModelized.py

import pylab as pl
import inspect
from types import *
from copy import copy, deepcopy
from ObjectModelized import *
import logging
import sys

logger = logging.getLogger("CarCostSimulator")

class Drivertype(ListManaged, XmlExp):

    def __init__(self, drivertypename = "builder"):
        ListManaged.__init__(self, drivertypename)
        self.drivingcoefficient = 1.0

    @property
    def drivingcoefficient(self):
        return self._drivingcoefficient
    @drivingcoefficient.setter
    def drivingcoefficient(self, value):
        self._drivingcoefficient = float(value)

class Driver(ListManaged, XmlExp):

    def __init__(self, driverName = "builder"):
        ListManaged.__init__(self, driverName)
        self.kmperyear = 0.0
        self.maxkilometers = 0.0
        self.drivertype = ""

    @property
    def drivertype(self):
        if self._drivertype is None:
            return Drivertype()
        elif type(self._drivertype) is not Drivertype:
            self.drivertype = self._drivertype
        return self._drivertype
```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
@drivertype.setter
def drivertype(self, value):
    try:
        if hasattr(Drivertype, 'arrayObj')
            and Drivertype.arrayObj.has_key(str(value)):
            self._drivertype = Drivertype.arrayObj[str(value)]
        else:
            self._drivertype = value
    except:
        self._drivertype = value
        logger.warning("Driver.drivertype setter : "
            + str(sys.exc_info()))

@drivertype.deleter
def drivertype(self):
    self._drivertype = None

@property
def kmperyear(self):
    return self._kmPerYear
@kmperyear.setter
def kmperyear(self, value):
    self._kmPerYear = float(value)

@property
def maxkilometers(self):
    return self._maxkilometers
@maxkilometers.setter
def maxkilometers(self, value):
    self._maxkilometers = float(value)

#do for unit tests
if __name__ == '__main__':
    pass
```

Finissons les classes métier par la gestion du carburant.

```
# Module FuelModelized.py

import pylab as pl
import inspect
from types import *
from copy import copy, deepcopy
from ObjectModelized import *
import logging

logger = logging.getLogger("CarCostSimulator")

class Fuel(ListManaged, XmlExp):
    """
    Class Fuel simulant les differents Fuels possibles
    """

    def __init__(self, fuelName = "NA", price = 0.0):
        ListManaged.__init__(self, fuelName)
        _price = 0.0
        self.price = price

    def __copy__(self):
        myClone = Fuel()
        myClone.name = self.name
        myClone.price = self.price
        return myClone

    def __deepcopy__(self, Memo = None):
        myClone = Fuel()
        myClone.name = self.name
        myClone.price = self.price
        return myClone

    @property
    def price(self):
        return self._price
    @price.setter
    def price(self, value):
        self._price = float(value)

    def getInfos(self):
        return (str(self) + ";" + str(self.price))
```

Analysez (vos besoins), Etudiez, Scriptez
L'art d'utiliser du scripting pour un projet

Maintenant, orientons nous vers la partie graphique. L'interface a été conçue avec Glade. Cela permet, de manière graphique (avec l'outil « Glade UI Designer ») de définir notre interface et d'exporter la description visuelle du programme en un fichier xml. Ce fichier xml sera par la suite intégré au programme principal.

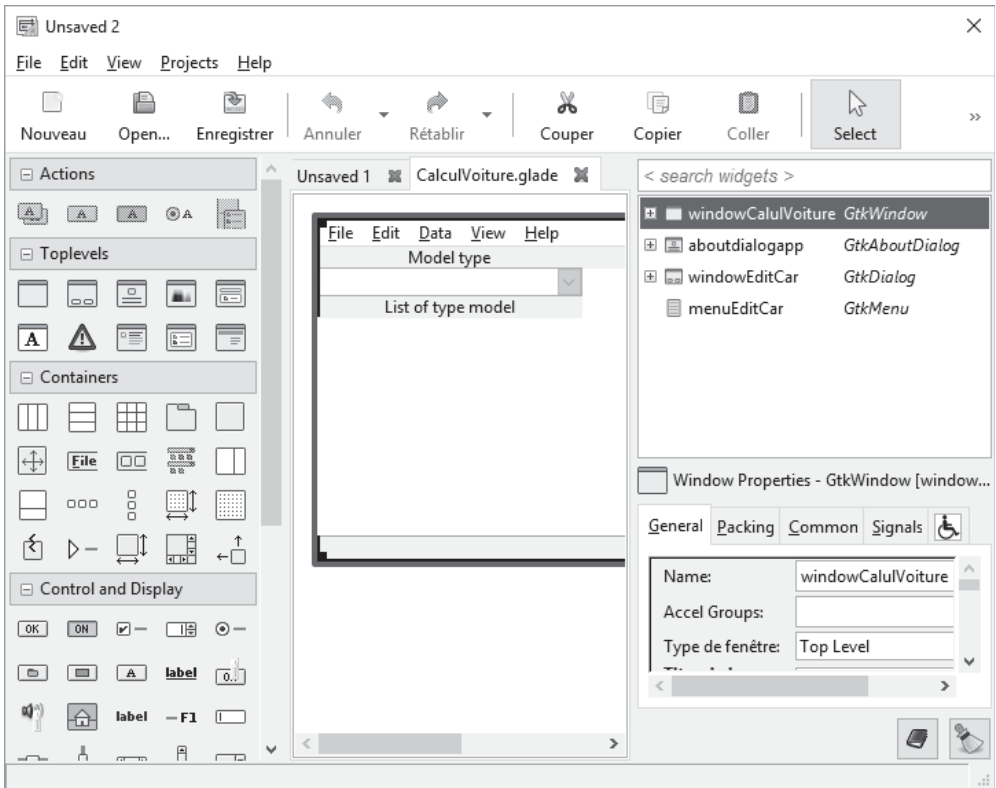


Illustration 14: Fenêtre principale de l'application sous Glade UI

Trois parties sont présentes dans cette interface:

- les boîtes à outils à gauche
- le sélecteur de fenêtres à droite.
- la fenêtre de dessin choisie dans le sélecteur, au centre.

Nous comprenons, par ces deux dernières parties, que le dessin de plusieurs fenêtres graphiques peut être compilé dans un seul fichier XML.

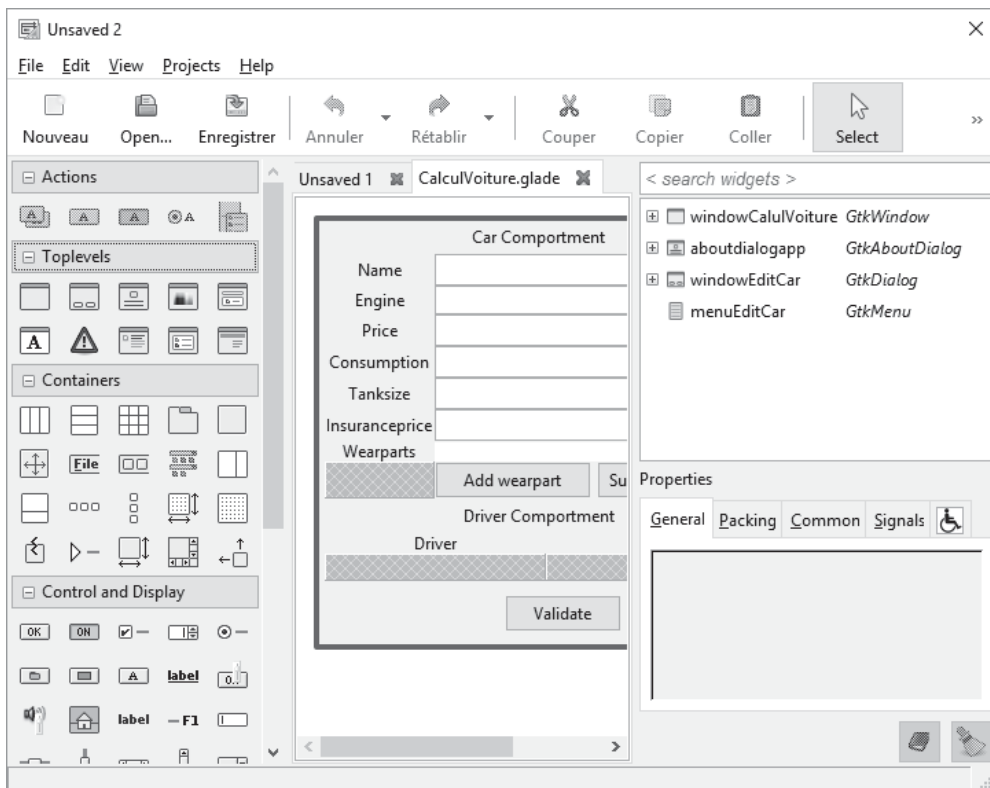


Illustration 15: Pop-up créée sous Glade UI

Voici un extrait du code généré par Glade. Nous pouvons constater que c'est du XML pur en version 1.0. Ouvert et facile à maintenir, ce dernier ne requiert aucun IDE pour le transformer ou l'adapter en cas de nouvelles demandes d'utilisateurs.

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
<?xml version="1.0"?>
<glade-interface>
  <!-- interface-requires gtk+ 2.16 -->
  <!-- interface-naming-policy project-wide -->
  <widget class="GtkWindow" id="windowCalulVoiture">
    <property name="visible">True</property>
    <property name="events"></property>
    <property name="title" translatable="yes">Calcul co&#x26;#x26;t
voiture</property>
    <property name="role">mainWindow</property>
    <property name="default_width">800</property>
    <property name="default_height">360</property>
    <child>
      <widget class="GtkVBox" id="vbox1">
        <property name="visible">True</property>
        <property name="orientation">vertical</property>
[...
        <widget class="GtkButton" id="buttonCancelCar">
          <property name="label"
translatable="yes">Cancel</property>
          <property name="visible">True</property>
[...
          <signal name="clicked"
handler="on_buttonCancelCar_clicked"/>
          <accelerator key="Escape" signal="activate"/>
        </widget>
        <packing>
          <property name="expand">False</property>
          <property name="fill">False</property>
          <property name="position">1</property>
        </packing>
      </child>
    </widget>
    <packing>
      <property name="expand">False</property>
      <property name="pack_type">end</property>
      <property name="position">5</property>
    </packing>
  </child>
</widget>
<child>
  <widget class="GtkMenu" id="menuEditCar">
    <property name="visible">True</property>
  </widget>
</glade-interface>
```

Après avoir défini chaque brique de notre programme, nous devons lier l'interface graphique au code métier.

Pour cela, définissons une classe WinInterface puis appelons le main Gtk qui s'occupera de faire vivre l'application.

```
#!/usr/bin/env python

import pygtk
pygtk.require('2.0')
import gtk
import gtk.glade
import gobject
from datetime import date
from copy import *
from types import *

import os
import os.path as path
import sys
from subprocess import call, Popen

import inspect

#Get abs path about this file
absFilePath = path.abspath('__file__')
#Get abs directory about this file
absFileDirectoryPath = path.dirname(absFilePath)
#Join into this directory sub module directory
modulesPyc = path.join(absFileDirectoryPath, 'modules')
#Calcul HMI path
hmiPath = path.join(absFileDirectoryPath, 'HMI',
'CalculVoiture.glade')

#Default data directory
dataDir = path.normpath(path.join(absFileDirectoryPath, '..', 'data'))

#import owns application's modules
sys.path.append(modulesPyc)

import logging
import logging.config

logFile = "traceback_20111023.txt"
logConfigFile = 'logging_basil.conf'
```


Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
#logging.config.fileConfig(logConfigFile)

loggermode = logging.INFO

#create logger
logger = logging.getLogger("CarCostSimulator")
logger.setLevel(loggermode)
#create console handler and set level to debug
ch = logging.StreamHandler()
ch.setLevel(loggermode)
#create formatter
formatter = logging.Formatter("%(asctime)s - %(name)s - %(
levelname)s - %(message)s")
#add formatter to ch
ch.setFormatter(formatter)
#add ch to logger
logger.addHandler(ch)

#"application" code
#logger.debug("debug message")
#logger.info("info message")
#logger.warn("warn message")
#logger.error("error message")
#logger.critical("critical message")

#import zipfile module if exist
zipmodule = path.join(modulesPyc, 'modelizing.zip')
if path.exists(zipmodule):
    sys.path.append(zipmodule)

from Modelizing import *
from XmlCarManage import *
```

```

class winInterface(object):

    def __init__(self, datafilename = ""):
        global hmiPath
        self._tmpObjCopy = None
        self._tmpObjEdit = None
        self._datafilename = datafilename
        self.widgets = gtk.glade.XML(hmiPath)
        self.autoConnect()
        self.createTreeViews()
        self.dataLoader()
        self._treeviewselected = None

    def __getitem__(self, widgetName):
        return self.widgets.get_widget(widgetName)

    def autoConnect(self):
        events = {}
        for (itemName, value) in self.__class__.__dict__.items():
            if callable(value) and itemName.startswith('gtk_'):
                events[itemName[4:]] = getattr(self, itemName)
        self.widgets.signal_autoconnect(events)
        # main frame managment
        self['windowCalulVoiture'].connect('destroy',
                                           self.gtk_main_quit)
        self['windowCalulVoiture'].connect('delete-event',
                                           self.delete_event)
        self['windowEditCar'].connect('delete-event',
                                      self.delete_event)

    def createTreeViews(self):
        self.createTreeViewByName(Wearpart, 'wearparttreeview',
                                  True, [])
        self.createTreeViewByName(CostKm, 'coststreeview', False, [])

        # List of fuel, driver, classes car
        self.createModelForCombobox('comboboxFuel',
                                    self.renderer_print_ObjName)
        self.createModelForCombobox('comboboxDriver',
                                    self.renderer_print_ObjName)
        self.createModelForCombobox('objcombobox',
                                    self.renderer_print_ClassName)

        self.fillObjComboBox()
        return

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
def createModelForCombobox(self, cbname, renderer_printing):
    if not self[cbname]: return
    self[cbname].set_model(gtk.ListStore(gobject.TYPE_PYOBJECT))
    cell = gtk.CellRendererText()
    self[cbname].pack_start(cell, True)
    self[cbname].set_cell_data_func(cell, renderer_printing)

def fillObjComboBox(self):
    liststore = self['objcombobox'].get_model()
    if len(liststore) > 0: liststore.clear()

    for classObj in listClassSimulator:
        liststore.append([classObj])

def alpha_or_num_sort(self, treemodel, iter1, iter2,
                      userData = None):
    try:
        item1 = treemodel[iter1][0]
        item2 = treemodel[iter2][0]
        return cmp(item1, item2)
    except:
        return 0
    return

def append_Column_Combobox_Into_TreeView(self, treeviewname,
                                          idx, columnName, listElems):
    cell_renderer = gtk.CellRendererCombo()
    cell_renderer.set_property('editable', True)
    cell_renderer.set_property('has-entry', False)

    # append list choice elements
    modelCB = gtk.ListStore(str)
    for elem in listElems: modelCB.append([str(elem)])
    cell_renderer.set_property('model', modelCB)
    cell_renderer.set_property('text-column', 0)

    cell_renderer.connect('edited', self.edited_cb,
                          (self[treeviewname], columnName))

    column = gtk.TreeViewColumn(columnName, cell_renderer)
    column.set_cell_data_func(cell_renderer,
                              self.renderer_print_data, self[treeviewname])

    self[treeviewname].insert_column(column, idx)
    return
```

```

def createTreeViewByName(self, treeviewobject,
                        treeviewname = "", editable = True, objColName = []):
    objName = treeviewobject.__name__
    if treeviewname == "":
        treeviewname = objName.lower() + 'treeview'

    liststore = gtk.ListStore(object)
    self[treeviewname].connect('state-changed',
                               self.widget_state_changed)
    self[treeviewname].set_model(liststore)
    #liststore.connect('row-deleted',
    #                  eval('self.row' + treeviewname + '_deleted'))
    #liststore.connect('row-changed',
    #                  eval('self.row' + treeviewname + '_changed'))
    #liststore.connect('row-inserted',
    #                  self.rowobjtreeview inserted)
    #liststore.connect('row-has-child-toggled', self.doAnything)
    #liststore.connect('rows-reordered', self.doAnything)

    # define the sortable condition for the ListStore
    liststore.set_sort_func(0, self.alpha_or_num_sort, None)
    liststore.set_default_sort_func(self.alpha_or_num_sort)

    # delete all of columns if exists
    columnsList = self[treeviewname].get_columns()
    if len(columnsList) > 0:
        for colObj in columnsList:
            self[treeviewname].remove_column(colObj)
    # delete items if exist
    self[treeviewname].get_model().clear()

    # Fill column's cells
    if len(objColName) == 0 :
        tmpMembers = inspect.getmembers(treeviewobject)
        for mbr in tmpMembers:
            propertyType = type(mbr[1])
            propertyName = mbr[0]

            # if the property is not a list of contents
            if propertyType.__name__ == "property":
                if propertyName.lower() == "name":
                    objColName.insert(0, propertyName)
                else:
                    objColName.append(propertyName)

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
for colName in objColName:
    cell_renderer = gtk.CellRendererText()
    cell_renderer.set_property('editable', editable)
    #if editable:
    cell_renderer.connect('edited', self.edited_cb,
                          (self[treeviewname], colName))
    column = gtk.TreeViewColumn(colName.title(),
                                cell_renderer)
    column.set_cell_data_func(cell_renderer,
                              self.renderer_print_data, self[treeviewname])
    self[treeviewname].append_column(column)
return

def widget_state_changed(self, source = None, event=None):
    logger.debug("def widget_state_changed " + str(source))

def renderer_print_data(self, column, cell_renderer, mod, itr,
                        user_data = None):
    #logger.debug("renderer_print_data " + str(column) + " "
    #            + str(cell_renderer) + " " + str(mod) + " "
    #            + str(itr) + " " + str(user_data))
    obj = mod[itr][0]
    if not obj: return

    if hasattr(mod[itr][0], column.get_title().lower()):
        cell_renderer.set_property('text', getattr(mod[itr][0],
                                                    column.get_title().lower()))
    else:
        if hasattr(mod[itr][0], "get" + column.get_title()):
            eval('self.renderer_print_Get' + column.get_title()
                + '(column, cell_renderer, mod, itr, user_data)')
    return

def renderer_print_GetInfos(self, column, cell_renderer, mod,
                            itr, user_data = None):
    if mod[itr][0]:
        cell_renderer.set_property('text',
                                    mod[itr][0].getInfos())
    return

def renderer_print_ObjName(self, column, cell_renderer, mod,
                            itr, user_data = None):
    if mod[itr][0]:
        cell_renderer.set_property('text', mod[itr][0].name)
    return
```

```

def render_print_ClassName(self, column, cell_renderer, mod,
                           itr, user_data = None):
    if mod[itr][0]:
        cell_renderer.set_property('text', mod[itr][0].__name__)
    return

def doAnything(self, cell = None, path = -1, new_text = "",
               user_data = None):
    #logger.debug("doAnything " + str(cell) + " " + str(path)
    # + " " + str(new_text) + " " + str(user_data))
    pass

def edited_cb(self, cell, path, new_text, user_data = None):
    if user_data == None:
        return
    try:
        treeview, propertyName = user_data
        # Get object
        tmpObj = treeview.get_model()[path][0]
        # put new value into propertyName's object
        setattr(tmpObj, propertyName.lower(), new_text)
    except:
        logger.warning("def edited_cb() : "
                       + str(sys.exc_info()))
    return

def fillObjTreeView(self, listObj, treeviewname):
    if not self[treeviewname]:
        logger.info("def fillObjTreeView: no '"
                    + treeviewname + "' declared")
        return

    logger.debug("def fillObjTreeView('" + str(listObj)
                  + "', " + treeviewname + ")")
    modelLS = self[treeviewname].get_model()
    modelLS.clear()

    for itm in listObj: modelLS.append([itm])
    modelLS.set_sort_column_id(0, gtk.SORT_ASCENDING)
    return

def gtk_newFile(self, source = None, event = None):
    self.clearAllData()
    self._datafilename = ""
    logger.info("New File")
    return

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
def clearAllData(self, clearObjects = True):
    # unselect objcombobox
    self['objcombobox'].set_active(-1)
    for objname in ['objtreeview', 'coststreeview']:
        modelLS = self[objname].get_model()
        if not modelLS is None:
            modelLS.clear()
    # clear objects
    if clearObjects == True:
        for objType in listClassSimulator:
            try:
                if hasattr(objType, 'clearObj'):
                    resp = objType.clearObj()
                    logger.debug("def clearAllData();
objType.clearObj(" + str(objType) + ") ==> " + str(resp))
            except:
                logger.warning("def clearAllData : "
                               + str(objType))

    self['labelCosts'].set_text("Costs")
    return

def gtk_openFile(self, source = None, event = None):
    global dataDir
    dialog = gtk.FileChooserDialog("Open ...", None,
        gtk.FILE_CHOOSER_ACTION_OPEN, (gtk.STOCK_CANCEL,
        gtk.RESPONSE_CANCEL, gtk.STOCK_OPEN, gtk.RESPONSE_OK))
    dialog.set_current_folder(dataDir)
    dialog.set_default_response(gtk.RESPONSE_OK)

    filter = gtk.FileFilter()
    filter.set_name("XmlData")
    filter.add_mime_type("data/xml")
    filter.add_pattern("*.xml")
    dialog.add_filter(filter)

    filter = gtk.FileFilter()
    filter.set_name("All files")
    filter.add_pattern("*")
    dialog.add_filter(filter)

    response = dialog.run()
    dialog.hide()
    if response == gtk.RESPONSE_OK:
        self.dataLoader(dialog.get_filename())
```

```
elif response == gtk.RESPONSE_CANCEL:
    logger.info('Closed, no files selected')
    dialog.destroy()
    return

def gtk_saveFile(self, source = None, event = None):
    if self._datafilename == "":
        self.gtk_saveAsFile()
    else:
        XmlCarManage.ExportDataToXML(self._datafilename)
        logger.info("File '" + self._datafilename + "' saved")
    return

def gtk_saveAsFile(self, source = None, event = None):
    global dataDir
    dialog = gtk.FileChooserDialog("Save ...", None,
        gtk.FILE_CHOOSER_ACTION_SAVE, (gtk.STOCK_CANCEL,
        gtk.RESPONSE_CANCEL, gtk.STOCK_SAVE, gtk.RESPONSE_OK))
    dialog.set_current_folder(dataDir)
    dialog.set_default_response(gtk.RESPONSE_OK)

    filter = gtk.FileFilter()
    filter.set_name("XmlData")
    filter.add_mime_type("data/xml")
    filter.add_pattern("*.xml")
    dialog.add_filter(filter)

    filter = gtk.FileFilter()
    filter.set_name("CsvData")
    filter.add_mime_type("data/csv")
    filter.add_pattern("*.csv")
    dialog.add_filter(filter)

    filter = gtk.FileFilter()
    filter.set_name("TxtData")
    filter.add_mime_type("data/txt")
    filter.add_pattern("*.txt")
    dialog.add_filter(filter)

    filter = gtk.FileFilter()
    filter.set_name("All files")
    filter.add_pattern("*")
    dialog.add_filter(filter)

    saveFile = False
    response = dialog.run()
```


Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
tmpFileName = ""
if response == gtk.RESPONSE_OK:
    tmpFileName = dialog.get_filename()

    if not pth.exists(tmpFileName):
        saveFile = True
    else:
        msgBox = gtk.MessageDialog(parent=None, flags=0,
                                    type=gtk.MESSAGE_QUESTION,
                                    buttons=gtk.BUTTONS_YES_NO, message_format=None)
        msgBox.set_markup("The file '" + tmpFileName
                           + "' already exists.\nDo you want to replace it ?")
        resp2 = msgBox.run()
        msgBox.hide()
        if resp2 == gtk.RESPONSE_YES:
            saveFile = True
        msgBox.destroy()
elif response == gtk.RESPONSE_CANCEL:
    logger.info('Closed, no files selected')
dialog.hide()
dialog.destroy()
# Exports XmlData to tmpFileName
if saveFile:
    self._datafilename = tmpFileName
    XmlCarManage.ExportDataToXML(self._datafilename)
    logger.info("File saved as '" + tmpFileName + "'")
return

def gtk_on_objcombobox_changed(self, source = None,
                                event = None, userData = None):
    logger.debug("def gtk_on_objcombobox_changed : "
                 + str(source))

    model = source.get_model()
    active = source.get_active()
    if active < 0:
        #clear Costs views
        try:
            self['labelCosts'].set_text("Costs")
            modelLS = self['coststreeview'].get_model()
            if not modelLS is None:
                modelLS.clear()
        except:
            pass
    return
```

```

classObj = model[active][0]

if classObj is Car:
    self.createTreeViewByName(classObj, "objtreeview",
                              False, ["Name", "Infos"])
elif classObj is Driver:
    self.createTreeViewByName(classObj, "objtreeview", True,
                              ["Name", "Kmperyear", "Maxkilometers"])
    if hasattr(Drivertype, "arrayObj"):
        self.append_Column_ComboBox_Into_TreeView(
            "objtreeview", 1, "Drivertype",
            Drivertype.arrayObj.values())
    else:
        self.append_Column_ComboBox_Into_TreeView(
            "objtreeview", 1, "Drivertype", [])
else:
    self.createTreeViewByName(classObj, "objtreeview",
                              True, [])
logger.debug("self.fillObjTreeView(classObj.arrayObj) : "
            + str(classObj))

if hasattr(classObj, 'arrayObj'):
    self.fillObjTreeView(classObj.arrayObj.values(),
                        "objtreeview")

return

def gtk_on_treeviews_row_activated(self, source = None,
                                   event = None, userData = None):
    # double clic
    selectingRow = source.get_selection()
    if not selectingRow:
        return
    model, iter = selectingRow.get_selected()
    obj = model[iter][0]

    if type(obj) is Car:
        # copy obj for work with rollback
        self._tmpObjEdit = deepcopy(obj)

        self.fillEditableCarWindow(self._tmpObjEdit)
        self['windowEditCar'].run()
        #self['windowEditCar'].hide()
        # update data model
        self.gtk_on_treeviews_cursor_changed(source, event,
                                              None)

    return

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
def fillEditableCarWindow(self, carSelected = None):
    # fill all properties
    tmpMembers = inspect.getmembers(carSelected.__class__)
    for mbr in tmpMembers:
        propertyType = type(mbr[1])
        propertyName = mbr[0]
        propertyObj = carSelected.__getattr__(propertyName)

        # if the property is not a list of contents
        if propertyType.__name__ == "property" and
            type(propertyObj) is not ListType:
            entryName = 'entry' + str(propertyName).title()
            if self[entryName] is not None:
                self[entryName].set_text(str(propertyObj))

    # fill wearpart treeview
    self.fillObjTreeView(self._tmpObjEdit.Wearparts,
                        'wearparttreeview')

    # fill the two combobox
    self.fillComboBoxObj(Fuel.arrayObj.values(), 'comboboxFuel')
    self.fillComboBoxObj(Driver.arrayObj.values(),
                        'comboboxDriver')

    # check into the fuel list witch kind of fuel is
    logger.debug("def fillEditableCarWindow "
                + str(type(carSelected.fuel)))
    self.activeItemCombobox(carSelected.fuel, 'comboboxFuel')
    self.activeItemCombobox(carSelected.driver,
                        'comboboxDriver')

    return

def activeItemCombobox(self, searchObj, comboboxname):
    getObj = lambda modelObj : str(modelObj[0])
    # get the name / object
    listObj = map(getObj, self[comboboxname].get_model())
    searchObj = str(searchObj)
    if searchObj in listObj:
        self[comboboxname].set_active(listObj.index(searchObj))
    else:
        logger.info("def activeItemCombobox: " + searchObj
                    + " not found into " + comboboxname)
    return
```

```

def fillComboBoxObj(self, listObj, comboboxname):
    model = self[comboboxname].get_model()
    model.clear()
    for itm in listObj: model.append([itm])
    return

#def gtk_on_objtreeview_columns_changed(self, source = None,
#                                     event = None):
#    print "def gtk_on_objtreeview_columns_changed("
#        + str(source) + ")"
#    return

def gtk_on_buttonValidateCar_clicked(self, source = None,
                                    event = None):
    selectingRow = self._treeviewselected.get_selection()
    model, iter = selectingRow.get_selected()
    obj = model[iter][0]

    if type(obj) is not Car:
        logger.info("def gtk_on_buttonValidateCar_clicked(); " \
                    "No Car type for object : '" + str(obj) + "'")
        return

    # get all members in Car class
    tmpMembers = inspect.getmembers(self._tmpObjEdit.__class__)
    for mbr in tmpMembers:
        propertyType = type(mbr[1])
        propertyName = mbr[0]
        propertyObj =
            self._tmpObjEdit.__getattr__(propertyName)

        # if the property is not a list of contents
        if propertyType.__name__ == "property"
            and type(propertyObj) is not ListType:
                entryName = 'entry' + str(propertyName).title()
                if self[entryName] is not None:
                    if hasattr(self._tmpObjEdit, propertyName):
                        setattr(self._tmpObjEdit, propertyName,
                                self[entryName].get_text())

    # get Fuel type and driver type
    tpNames = ['fuel', 'driver']
    for tpName in tpNames:
        cbName = 'combobox' + tpName.title()
        tmpmodel = self[cbName].get_model()
        iterActiv = self[cbName].get_active_iter()

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
        if iterActiv:
            setattr(self._tmpObjEdit, tpName,
                    str(tmpmodel[iterActiv][0]))

    # refresh calc cost
    self._tmpObjEdit.refreshCalc()
    # replace Car object into Car list
    Car.updateObj(obj, self._tmpObjEdit)

    # replace Car object into treeview
    #model.set_value(iter, 0, self._tmpObjEdit)
    # refresh treeview
    self.gtk_refreshData(None, None)

    self._tmpObjEdit = None
    self['windowEditCar'].hide()
    return

#def rowobjtreeview_changed(self, source = None,
#                             event = None, userData = None):
#    #logger.debug("rowobjtreeview_changed " + str(source)
#    #              + " " + str(event) + " " + str(userData))
#    pass

#def rowobjtreeview_inserted(self, source = None,
#                             event = None, userData = None):
#    logger.debug("rowobjtreeview_inserted " + str(source)
#    #              + " " + str(event) + " " + str(userData))
#    return

#def rowobjtreeview_deleted(self, source = None,
#                             event = None, userData = None):
#    #logger.debug("rowobjtreeview_deleted " + str(source)
#    #              + " " + str(event) + " " + str(userData))
#    pass

def gtk_on_buttonAddWearpart_clicked(self, source = None,
                                     event = None, userData = None):
    if userData == None:
        userData = Wearpart("newObj")

    self._tmpObjEdit.Wearparts.append(userData)

    modelWp = self['wearparttreeview'].get_model()
    modelWp.append([userData])
```

```

logger.debug("def gtk_on_buttonAddWearpart_clicked")
return

def gtk_on_buttonSuppressWearpart_clicked(self, source = None,
                                         event = None, userData = None):
    rowselecting = self['wearparttreeview'].get_selection()
    model, iter = rowselecting.get_selected()

    objWearpart = model[iter][0]
    wpName = objWearpart.name
    idxWp = self._tmpObjEdit.Wearparts.index(objWearpart)
    if idxWp >= 0:
        del self._tmpObjEdit.Wearparts[idxWp]
        model.remove(iter)
        logger.info("Suppress Wearpart '" + wpName + "'")

    if self["buttonSuppressWearpart"]:
        self["buttonSuppressWearpart"].set_property('sensitive',
                                                    False)

    return

def gtk_on_treeviews_cursor_changed(self, source = None,
                                    event = None, userData = None):
    # data selected
    logger.debug("cursor_changed " + str(source) + " "
                + str(event) + " " + str(userData))
    self._treeviewsselected = source
    selectingRow = source.get_selection()
    model, iter = selectingRow.get_selected()
    if not model or not iter:
        return
    obj = model[iter][0]
    if type(obj) is Car:
        # Get the new tmpObj
        self._tmpObjEdit = obj
        self.gtk_showGraph(source, event, obj)
        # view costs
        self.viewCosts(obj)

    # call event to activate edit menu
    self.activeEditMenu()
    return

def gtk_on_wearparttreeview_cursor_changed(self, source = None,
                                           event = None, userData = None):
    if self["buttonSuppressWearpart"]:

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
        self["buttonSuppressWearpart"].set_property('sensitive',
                                                    True)
    return

def activeEditMenu(self):
    self.manageEditMenu(True)
    return

def desactiveEditMenu(self):
    self.manageEditMenu(False)
    return

def manageEditMenu(self, sensitiveValue = False):
    listEditableMenu = ["menuitemsuppress", "menuitemcopy",
                        "menuitemcut"]
    # check if there is anything to paste
    # if there is it, please activate paste item menu
    if self["menuitempaste"]:
        self["menuitempaste"].set_property('sensitive',
                                            self._tmpObjCopy is not None)

    for tmpMenu in listEditableMenu:
        if self[tmpMenu]:
            self[tmpMenu].set_property('sensitive',
                                       sensitiveValue)

    return

def gtk_main_quit(self, source = None, event = None):
    gtk.main_quit()

def gtk_on_buttonCancelCar_clicked(self, source = None,
                                   event=None):
    if self['windowEditCar'] is not None:
        self['windowEditCar'].hide()

    if self._tmpObjCopy:
        self._tmpObjCopy = None

    return

def delete_event(self, source = None, event=None):
    # don't destroy window -- just leave it hidden
    # for now .. fix later
    source.hide()
    return
```

```

def viewCosts(self, objCar):
    logger.debug("def viewCosts")
    x_tab = range(0, Car.maxkilometers, Car.espaceKm)
    y_tab = objCar.Costs

    CostKm.arrayObj = []

    for idx in range(0, len(x_tab), 1):
        lgn = CostKm()
        lgn.km = x_tab[idx]
        lgn.price = y_tab[idx]
        CostKm.arrayObj.append(lgn)

    self['labelCosts'].set_text(objCar.name)
    self.fillObjTreeView(CostKm.arrayObj, "coststreeview")
    return

def gtk_showGraph(self, source = None, event = None,
                  UserData = None):
    if type(UserData) is not Car:
        return

    self['carcurve'].reset()
    self['carcurve'].set_range(0, Car.maxkilometers, 0, 50000)
    self['carcurve'].set_curve_type(gtk.CURVE_TYPE_FREE)
    self['carcurve'].set_vector(UserData.Costs)

    return

def gtk_saveCalcul(self, source = None, event = None):
    global dataDir
    tmpPath = 'compCars_' + date.today().strftime("%Y%m%d")
        + '.csv'
    tmpPath = pth.join(dataDir, 'results', tmpPath)
    CarModelized.sauvComparaisonCSV(tmpPath)
    self.processViewingFile(tmpPath)
    logger.info("Calculs saved")
    return

def gtk_saveArrayText(self, source = None, event = None):
    global dataDir
    tmpPath = 'compCars_' + date.today().strftime("%Y%m%d")
        + '.txt'
    tmpPath = pth.join(dataDir, 'results', tmpPath)

```


Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
CarModelized.sauvComparaisonText(tmpPath)
self.processViewingFile(tmpPath)
logger.info("Array Text saved")
return

def gtk_refreshData(self, source = None, event = None):
    # send new car calcul
    logger.debug("refresh data")
    # clear all viewing data
    self.clearAllData(False)
    # reload the first category
    self['objcombobox'].set_active(0)
    return

def gtk_reloadData(self, source = None, event = None):
    msgBox = gtk.MessageDialog(parent=None, flags=0,
                               type=gtk.MESSAGE_QUESTION, buttons=gtk.BUTTONS_YES_NO,
                               message_format=None)
    msgBox.set_markup("Do you really want to reload the file '"
                      + self._datafilename
                      + "'. It will errase your actually work ?")
    reloadFile = msgBox.run() == gtk.RESPONSE_YES
    msgBox.hide()
    msgBox.destroy()

    if reloadFile:
        self.dataLoader()
        logger.info("Data reloaded")
    return

def gtk_showCompared(self, source = None, event = None):
    if CarModelized.showCurves() == 0:
        msgBox = gtk.MessageDialog(parent=None, flags=0,
                                    type=gtk.MESSAGE_INFO, buttons=gtk.BUTTONS_OK,
                                    message_format=None)
        msgBox.set_markup('No data present !')
        msgBox.run()
        msgBox.hide()
        msgBox.destroy()
    return

def gtk_showAbout(self, source = None, event = None):
    if self['aboutdialogapp'] != None:
        self['aboutdialogapp'].run()
        self['aboutdialogapp'].hide()
```

```

else:
    logger.warning("Error creating widget about for "
                  + "'def gtk_showAbout'")
    return

def gtk_dataAppendNewObj(self, source = None, event = None,
                        userData = None):
    objType = None
    idxType = 0
    try:
        idxType = self['objcombobox'].get_active()
        modelType = self['objcombobox'].get_model()
        objType = modelType[idxType][0]
    except:
        logger.warning("error on getting object type in "
                      + "gtk_dataAppendNewObj function")
        return

    # if there is no Type select, says it by pop-up
    if idxType < 0:
        # pop-up
        msgBox = gtk.MessageDialog(parent=None, flags=0,
                                   type=gtk.MESSAGE_INFO, buttons=gtk.BUTTONS_OK,
                                   message_format=None)
        msgBox.set_markup("Please, selecte one category "
                          + "before inserted object")
        msgBox.run()
        msgBox.hide()
        msgBox.destroy()
        logger.info("error on inserting new object "
                   + "without type model selected")
        return

    if userData is None:
        userData = objType("newObj")
        logger.debug("def gtk_dataAppendNewObj "
                   + str(objType) + '("newObj")')

    # append new data into class array
    objType.appendObj(userData)
    # refresh printed data
    self.gtk_on_objcombobox_changed(self['objcombobox'], None,
                                    None)
    self['objcombobox'].set_active(idxType)

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
# select the new value
getObj = lambda modelObj : str(modelObj[0])
# get the name / object
listObj = map(getObj, self['objtreeview'].get_model())
self['objtreeview'].set_cursor(listObj.index(str(userData)))

self['objtreeview'].get_model().set_default_sort_func(
    self.alpha_or_num_sort)

#gtk_on_treeviews_cursor_changed
logger.debug("def gtk_dataAppendNewObj append new "
             + userData.__class__.__name__.title())
return

def gtk_closeAbout(self, source = None, event = None):
    self['aboutdialogapp'].hide()
    return

def gtk_dataCopy(self, source = None, event = None):
    selectingRow = self._treeviewselected.get_selection()
    result = selectingRow.get_selected()

    if result: #result could be None
        model, iter = result
        self._tmpObjCopy = model[iter][0]
        #self._tmpObjCopy.name += " (cpy)"

    logger.info("dataCopy " + str(self._tmpObjCopy))

    self.activeEditMenu()
    return

def gtk_dataPaste(self, source = None, event = None):
    if self._tmpObjCopy is None:
        return

    tmpCpy = copy(self._tmpObjCopy)
    tmpCpy.name += " (cpy)"

    self.gtk_dataAppendNewObj(None, None, tmpCpy)
    logger.info("dataPaste " + str(tmpCpy))
    return
```

```

def gtk_dataCut(self, source = None, event = None):
    self.gtk_dataCopy(source, event)
    self.gtk_dataSuppress(source, event)
    logger.info("dataCut to tmp : " + str(self._tmpObjCopy))
    return

def gtk_dataSuppress(self, source = None, event = None):
    if self._treeviewselected == None:
        return

    selectingRow = self._treeviewselected.get_selection()
    result = selectingRow.get_selected()

    if result: #result could be None
        model, iter = result

        objToRemove = model[iter][0]
        model.remove(iter)
        objToRemove.__class__.removeObj(objToRemove)
        self.desactiveEditMenu()

    return

def dataLoader(self, datafilename = ""):
    # test filename
    if datafilename != "":
        self._datafilename = datafilename

    if self._datafilename == "" or
       not path.exists(self._datafilename):
        self._datafilename = ""
        return

    # clear all data
    self.clearAllData()
    # load data contented in an other file
    XmlCarManage(self._datafilename)
    # select the first class into objcombobox
    self['objcombobox'].set_active(0)

    CarModelized.printCarsCompaired()
    Car.calcTabDiffAmortissements()
    return

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

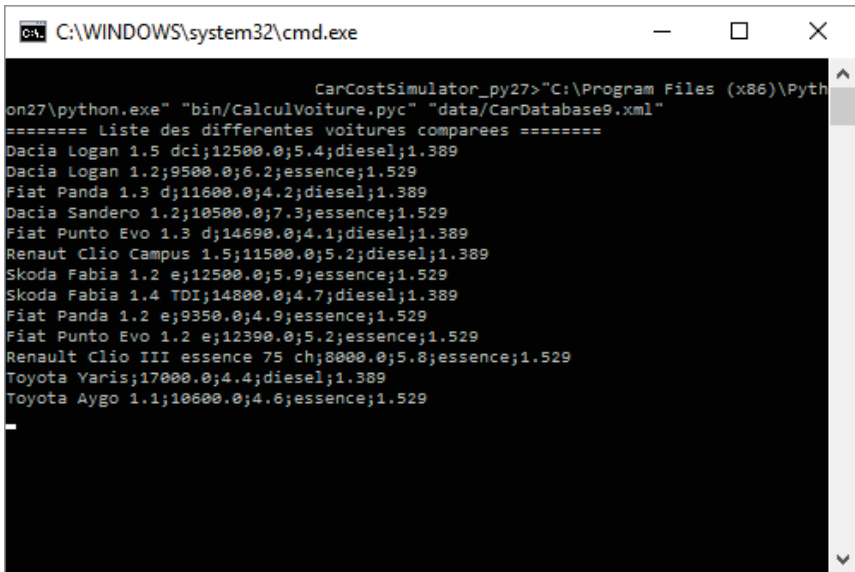
```
def processViewingFile(self, filename):
    os.system('"' + filename + '"')
    #Popen('"' + filename + '"', bufsize=0, executable=None,
    #      stdin=None, stdout=None, stderr=None,
    #      preexec_fn=None, close_fds=False,
    #      shell=False, cwd=None, env=None,
    #      universal_newlines=False,
    #      startupinfo=None, creationflags=0)

### main

if __name__ == '__main__':
    fileName = ""
    if len(sys.argv) > 1:
        fileName = sys.argv[1]
    mainWindow = winInterface(fileName)
    gtk.main()
```

Pour lancer le programme, il suffit de créer un fichier de démarrage ".sh", ".ps1", ".bat", suivant votre système d'exploitation.

Le but étant d'appeler l'interpréteur Python (version 2.6 ou 2.7, car toutes les bibliothèques utilisées n'ont pas été migrées sous Python 3.x), en lui passant en paramètre le nom du programme principal (CalculVoiture.py). Nous verrons par la suite que les programmes Python peuvent être compilés et être déployés sans code source sur les machines cibles (comme Java et DotNet).



```
C:\WINDOWS\system32\cmd.exe

CarCostSimulator_py27>"C:\Program Files (x86)\Python27\python.exe" "bin/CalculVoiture.pyc" "data/CarDatabase9.xml"
==== Liste des différentes voitures comparees =====
Dacia Logan 1.5 dci;12500.0;5.4;diesel;1.389
Dacia Logan 1.2;9500.0;6.2;essence;1.529
Fiat Panda 1.3 d;11600.0;4.2;diesel;1.389
Dacia Sandero 1.2;10500.0;7.3;essence;1.529
Fiat Punto Evo 1.3 d;14600.0;4.1;diesel;1.389
Renaut Clio Campus 1.5;11500.0;5.2;diesel;1.389
Skoda Fabia 1.2 e;12500.0;5.9;essence;1.529
Skoda Fabia 1.4 TDI;14800.0;4.7;diesel;1.389
Fiat Panda 1.2 e;9350.0;4.9;essence;1.529
Fiat Punto Evo 1.2 e;12390.0;5.2;essence;1.529
Renault Clio III essence 75 ch;8000.0;5.8;essence;1.529
Toyota Yaris;17000.0;4.4;diesel;1.389
Toyota Aygo 1.1;10600.0;4.6;essence;1.529
```

Illustration 16: Résultat console du chargement des différents véhicules renseignés

Analysez (vos besoins), Etudiez, Scriptez
L'art d'utiliser du scripting pour un projet

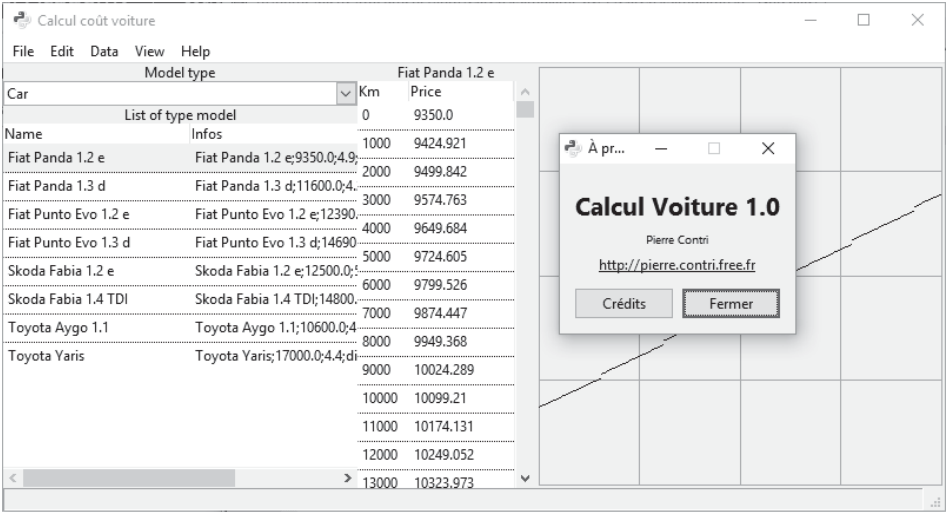


Illustration 17: Voici donc un aperçu du travail final en exécution

Un affichage de comparaison en mode texte sur un éditeur (Linux TextEdit ou Windows Notepad).

```

compCars_161028.txt - Bloc-notes
Fichier Edition Format Affichage ?

Matrice de comparaison
=====
| Fiat Panda 1.3 d          || Fiat Panda 1.3 d          || Fiat Punto Evo 1.3 d
| Fiat Panda 1.3 d          || 0                          || -1
| Fiat Punto Evo 1.3 d      || 1                          || 0
| Skoda Fabia 1.2 e         || 1                          || 87000
| Skoda Fabia 1.4 TDI       || 1                          || 1
| Fiat Panda 1.2 e          || -1                         || -1
| Fiat Punto Evo 1.2 e      || 1                          || -1
| Toyota Yaris              || 1                          || 1
| Toyota Aygo 1.1           || 180000                     || -1

Comparaison Voitures
=====
Fiat Panda 1.3 d          = Fiat Panda 1.3 d          ==> 0
Fiat Panda 1.3 d          < Fiat Punto Evo 1.3 d          ==> /
Fiat Panda 1.3 d          < Skoda Fabia 1.2 e              ==> /
Fiat Panda 1.3 d          < Skoda Fabia 1.4 TDI            ==> /
Fiat Panda 1.3 d          > Fiat Panda 1.2 e               ==> /
Fiat Panda 1.3 d          < Fiat Punto Evo 1.2 e           ==> /
Fiat Panda 1.3 d          < Toyota Yaris                   ==> /

```

Illustration 18: Résultat en fichier texte de la comparaison

Vous pourrez télécharger l'application complète sur CodePlex (<http://carcostsimulator.codeplex.com>).

Cela montre qu'avec Python, il est tout à fait possible de créer une application lourde et complexe, en langage dit de "Scripting". On pourra alors faire tomber tous les préjugés, et comprendre que tout bon langage de programmation peut être utilisé pour tous types d'applications, même si chacun a sa spécificité.

Javascript

Contenu de ce chapitre

- Javascript..... 137
 - Pourquoi utiliser ce langage..... 138
 - Afficher simplement 'Hello World'..... 139
 - Comment utiliser les scripts par la console..... 140
 - Aller plus loin avec le Scripting en mode console..... 141
 - Interaction entre le Scripting et une interface graphique..... 146
 - Le Pong..... 147
 - Le Pong2..... 150

Pourquoi utiliser ce langage

Créé au départ pour l'interaction entre deux pages web, ainsi que des animations sur les navigateurs, le langage JavaScript a conquis d'autres types de développeurs.

Actuellement, on peut utiliser JavaScript pour le Web au niveau « Front End », mais aussi pour des fonctions de « Back End », et de traitement de la donnée. Certaines personnes utilisent même JavaScript pour réaliser des scénarios de mise en production et déploiement de solutions informatiques complètes.

Afficher simplement 'Hello World'

```
document.write("hello world !");
```

Par défaut, vous serez obligé d'exécuter ce bout de code (unique dans un fichier) par l'intermédiaire d'un fichier html, afin de récupérer le rendu sur un navigateur.

Exemple d'un fichier permettant l'exécution du code ci-dessus:

```
<!DOCTYPE html>
<html>
  <head>
    <script type="text/javascript" src="./helloworld.js"></script>
  </head>

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

Résultat d'exécution:

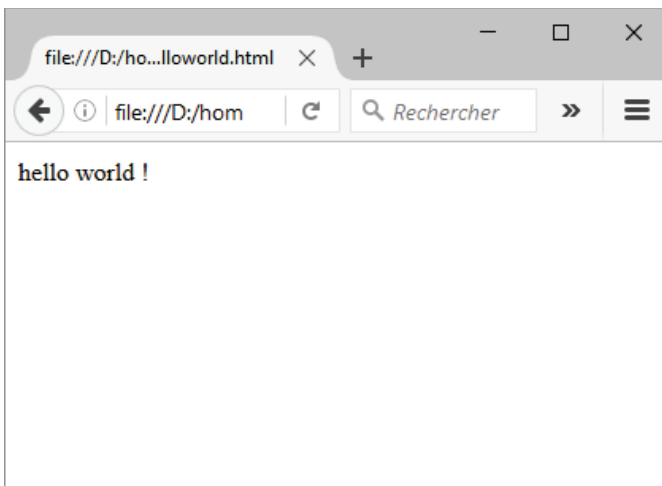


Illustration 19: Hello World en mode graphique

Comment utiliser les scripts par la console

Au départ, JavaScript n'était pas pensé pour être utilisé en mode console. Ce langage était à destination des navigateurs Web (tels que Netscape). Par la suite, ce langage a évolué et a été porté pour NodeJS.

NodeJS est un moteur intégrant un Framework qui permet d'interagir avec le système hôte. Ouvrir une Socket HTTP, lire un fichier sur le poste serveur, ... Grâce à ces fonctionnalités, il est possible d'écrire un site web grâce à cette technologie.

NodeJS est un exécutable que l'on peut télécharger gratuitement à travers le site officiel (<http://nodejs.org>).

Au vu des multiples versions présentes, j'ai choisi le téléchargement de l'exécutable seul, que je remplacerai manuellement lors de mises à jour. Après avoir enregistré le chemin de l'application dans la variable d'environnement PATH, nous pouvons lancer la console interactive de NodeJS, tel que:



Illustration 20: Hello World avec NodeJS

Maintenant, nous venons d'amener JavaScript au même niveau que PHP et Python, c'est à dire côté client lourd avec possibilité de réaliser des traitements plus poussés que simplement de l'affichage graphique.

Aller plus loin avec le Scripting en mode console

L'exemple ci-dessous est extrêmement simple. Il va lire un fichier CSV, le parcourir ligne par ligne et transformer chaque entrée en un objet spécifique.

Cet exemple a pour seul but de montrer que la programmation se fait de la même manière que PHP ou Python.

```
/* ***** */
/*      JavaScript Demo      */
/*      With NodeJS         */
/* ----- */
/* author   : Pierre Contri */
/* created  : 2016-12-04     */
/* ***** */

// using filesystem
fs = require('fs');

// open a csv document
// put it in dictionary
// return informations about object type

// class for software type definition
function SoftwareObject(objToParse) {

    // check the input parameter is an array >= 4 fields
    if (!(Array.isArray(objToParse) && objToParse.length > 3)) {
        return null;
    }

    // Columns description
    // column 1 id
    // column 2 file path
    // column 3 comments
    // column 4 subject

    this.id = objToParse[0].trim();
    this.path = objToParse[1].trim();
    this.comments = objToParse[2].trim();
    this.subject = objToParse[3].trim();
}
```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
SoftwareObject.prototype.toString = function () {
    return("{id: " + this.id + ", path: " + this.path + ", subject: "
        + this.subject + ", comments: " + this.comments + "}");
};

function getSoftwareObject(objToParse) {
    // check the input parameter is an array >= 4 fields
    if (!(Array.isArray(objToParse) && objToParse.length > 3))
        return undefined;

    return new SoftwareObject(objToParse);
}

function extractSubjectsFromSoftwareObjectList(softwareLst) {
    var subjectList = [];

    for(var i = 0; i < softwareLst.length; i++) {
        var subj = softwareLst[i].subject;
        if(subjectList.indexOf(subj) == -1)
            subjectList.push(subj);
    }

    return subjectList;
}

function extractDictSubjectNumbersFromSoftObjectList(softwareLst) {
    var dictSubject = {};
    for (var j = 0, len = softwareLst.length; j < len; j++) {
        var soft = softwareLst[j];

        if(! (soft.subject in dictSubject))
            dictSubject[soft.subject] = 1;
        else
            dictSubject[soft.subject] += 1;
    }
    return dictSubject;
}

//pads left
String.prototype.lpad = function(padString, length) {
    var str = this;
    while (str.length < length)
        str = padString + str;
    return str;
}
```

```
//pads right
String.prototype.rpad = function(padString, length) {
    var str = this;
    while (str.length < length)
        str = str + padString;
    return str;
}

function main() {

    // get the filename to open
    var filename = (process.argv.length > 2)
        ? process.argv[2] : "software.csv";

    // open the file if exist
    // if not, exit program
    if (!fs.existsSync(filename)) {
        console.log("The file '" + filename + "' does not exists\n");
        return;
    }

    var softwareList = [];

    filename = fs.realpathSync(filename);

    // load the file content to an array of softwareObject
    var textFile = fs.readFileSync(filename, 'utf-8');
    var lines = textFile.split(/\n|\r\n/);

    for(var i = 0; i < lines.length; i++) {
        var line = lines[i];
        // extract columns
        var objToParse = line.split("|");
        // create software object if the line is compatible
        var soft = getSoftwareObject(objToParse);

        if(soft != undefined)
            softwareList.push(soft);
    }

    // Extract all subjects from the software listStyleType
    var subjects =
        extractSubjectsFromSoftwareObjectList(softwareList);
```


Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
// Print keys name
console.log("\nKeys list\n-----");
for (var i = 0, len = subjects.length; i < len; i++) {
    console.log(subjects[i]);
}

// Count the number of entries per category
var dictSubject =
    extractDictSubjectNumbersFromSoftObjectList(softwareList);

// print informations for the dictionnary
console.log("\nCount number of using time per key"
    + "\n-----");
for (var k in dictSubject) {
    console.log(k.toString().rpad(" ", 30) + "|"
        + dictSubject[k].toString().lpad(" ", 4));
}

// Print sample lines of software informations
console.log("\nSample of Software List"
    + "\n-----");
for (var l = 0; l < 3; l++) {
    console.log(softwareList[l]);
}
}

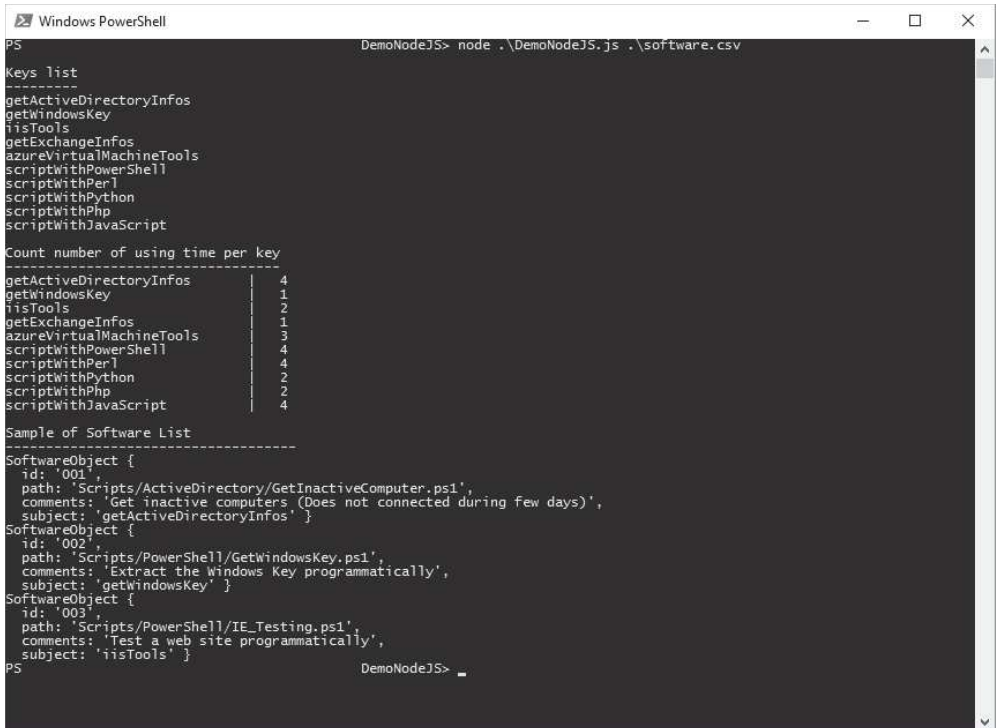
main();
```

Pour chaque ligne lu dans le fichier CSV, si cette dernière n'est ni vide ni commentée et qu'elle contient bien quatre colonnes, nous créerons un objet de type "SoftwareObject" en spécifiant les quatre colonnes à la création.

Le cours algorithme montre que l'on peut jouer avec la programmation multi paradigme en travaillant sur de la programmation procédurale et orientée objet.

J'ai volontairement mis la partie d'exécution principale dans une fonction "main" et l'ai appelée juste à la fin pour faire un clin d'œil à d'autres langages que j'apprécie, tel que le C, le C++. Java et C# ont aussi un « main » au démarrage.

Le résultat en image:



```
PS DemoNodeJS> node .\DemoNodeJS.js .\software.csv

Keys list
-----
getActiveDirectoryInfos
getWindowsKey
iisTools
getExchangeInfos
azureVirtualMachineTools
scriptWithPowerShell
scriptWithPerl
scriptWithPython
scriptWithPhp
scriptWithJavaScript

Count number of using time per key
-----
getActiveDirectoryInfos      4
getWindowsKey                1
iisTools                     2
getExchangeInfos            1
azureVirtualMachineTools     3
scriptWithPowerShell         4
scriptWithPerl               4
scriptWithPython             2
scriptWithPhp                2
scriptWithJavaScript         4

Sample of Software List
-----
SoftwareObject {
  id: '001',
  path: 'Scripts/ActiveDirectory/GetInactiveComputer.ps1',
  comments: 'Get inactive computers (Does not connected during few days)',
  subject: 'getActiveDirectoryInfos' }
SoftwareObject {
  id: '002',
  path: 'Scripts/PowerShell/GetWindowsKey.ps1',
  comments: 'Extract the Windows Key programmatically',
  subject: 'getWindowsKey' }
SoftwareObject {
  id: '003',
  path: 'Scripts/PowerShell/IE_Testing.ps1',
  comments: 'Test a web site programmatically',
  subject: 'iisTools' }
PS DemoNodeJS> _
```

Illustration 21: JavaScript en mode console avec NodeJS

Interaction entre le Scripting et une interface graphique

Depuis plus de vingt-cinq années, JavaScript permet de réaliser des interfaces graphiques en Web. Ce qui manque dans ce langage multi paradigme, c'est simplement l'imagination des différents utilisateurs.

Par exemple, en 2004, date à laquelle jQuery (un Framework JavaScript) n'existait pas, la majorité des gens pensaient qu'il n'était pas possible de créer un jeu animé sans allers-retours vers le serveur. Le moment où Flash était « LE » langage qu'il fallait absolument utiliser pour réaliser des animations internet.

Voici la preuve, que malgré un graphique très mauvais (car je ne suis pas graphiste), il était tout à fait possible de construire une application autonome.

Le Pong



Illustration 22: Premier mouvement de balle sur le Pong

Dans ce tout premier exemple, j'ai créé une boucle permettant de simuler une balle et une raquette, tout en code ASCII.

Ce pong rend hommage au tout premier jeu du Pong, réalisé 50 ans plus tôt. Cette démonstration ne se veut pas autant intéressante ni élégante que le tout premier pong, mais le but de ce code était de faire taire certains de mes collègues qui me soutenaient que ce n'était pas possible de l'écrire en JavaScript.

Voici le code, écrit en 2004 en 1/2 journée. Ne nous attardons pas sur sa qualité, ni quoi que ce soit, mais celui-ci fonctionne depuis plus de 12 ans maintenant, sans mise à jour ni aucun changement, et ce malgré le changement de navigateurs et de technologies satellites.

```
<html>
<head>
<style type="text/css">
.tablePingPong {
    border-style : solid;
    border-color : #000000;
    border-height : 0px;
    border-width : 0px 0px 0px 0px;
    padding : 1px 1px 1px 1px;
    height : 100%;
    width : 100%;
}
.trPingPong {
    border-style : solid;
    border-color : #000000;
    border-height : 0px;
    border-width : 0px;
    padding : 1px 1px 1px 1px;
    height : 9px;
}
```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
.tdPingPong {
    border-style : solid;
    border-height : 0px;
    border-width : 0px 0px 1px 0px;
    padding : 1px 1px 1px 1px;
    height : 9px;
}
.tdRaquette {
    border-style : solid;
    border-height : 0px;
    border-width : 0px 0px 0px 0px;
    padding : 1px 1px 1px 1px;
    height : 9px;
}
div.dping {
    position : relative;
    left : 0px;
    top : 0px;
    width : 2%;
}
div.dpong {
    position : relative;
    left : 0px;
    top : 0px;
    width : 2%;
}
</style>
<script language="javascript">
x = 0;
y = 0;
posB = 0;

function joue()
{
    elemPing = document.getElementById("divping");
    elemPong = document.getElementById("divpong");
    elemBalle = document.getElementById("divballeping");

    if (elemPing && elemPong && elemBalle)
    {
        elemPing.style.visibility = (posB < 40) ? "visible" : "hidden";
        elemPong.style.visibility = (posB > 140) ? "visible" : "hidden";
    }
}
```

```
function balle()
{
  if(x>90) x=0;
  if(y>90) y=0;

  elemBalle = document.getElementById("divballeping");
  if (elemBalle)
  {
    echelleX = document.body.clientWidth * 0.36;
    echelleY = 50 / document.body.clientHeight;
    echelleY = 1 / echelleY;
    posB = Math.abs(Math.round((Math.cos(x) + 1) * echelleX));
    elemBalle.style.left = posB;
    elemBalle.style.top = -Math.abs(Math.round((Math.sin(y)) * 20 *
echelleY));
    x++;
    y++;
  }
  joue();
  setTimeout('balle()',150);
}
</script>
</head>
<body onload="balle()"
onresize="javascript:document.location.reload();">
  <table id="idTable" class="tablePingPong">
    <tr id="trPing1" class="trPingPong" cellpadding="0"
cellspacing="0" valign="bottom">
      <td id="tdRack1" class="tdRaquette" width="10%"
align="right"><div id="divping" style="visibility:
hidden;"><O<br>&nbsp;&nbsp;&nbsp;</div></td>
      <td id="tdPing1" class="tdPingPong" width="39%"
align="left"><div id="divballeping" class="dping" style="visibility:
visible;"><O</div></td>
      <td id="tdFil" class="tdPingPong" width="2%" align="center">|
</td>
      <td id="tdPing2" class="tdPingPong" width="39%"
align="left"><div id="divballepong" class="dpong" style="visibility:
hidden;"><O</div></td>
      <td id="tdRack2" class="tdRaquette" width="10%"
align="left"><div id="divpong" style="visibility:
hidden;"><O<br>&nbsp;&nbsp;&nbsp;</div></td>
    </tr>
  </table>
</body>
</html>
```

Prenons maintenant cet exemple, et transformons cet exercice de simulation de balle pour l'adapter au magnifique jeu "Archanoïd" créé au début des années 80 sur Thomson et sur bien d'autres plate-formes (ex Amiga ou Atari, ...).

Le Pong2

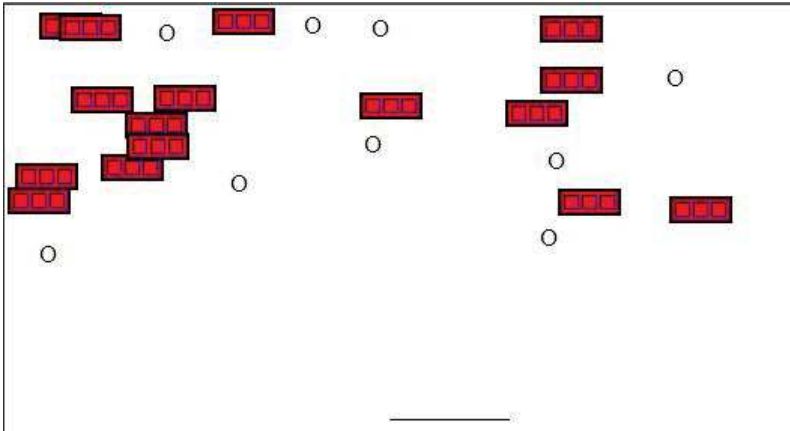


Illustration 23: Le Pong2 en JavaScript réalisé en 2006

Ce jeu consiste à casser les différentes briques (représentées par des tableaux HTML en rouge) avec des balles (représentées par une ou plusieurs lettres ASCII noire) en faisant rebondir ces dernières sur un plateau (représenté par plusieurs caractères soulignés-bas). Ce plateau répond aux touches gauche et droite, ainsi qu'au mouvement horizontal de la souris.

En 2005, ce jeu était écrit complètement en philosophie Procédurale. Par la suite, la majeure partie a été migrée en philosophie Orienté Objet. Dans un premier temps, cette transformation a été réalisée pour montrer que cela était possible, et aussi pour montrer que certains codes écrits en Orienté Objet étaient plus lents que ceux en Procédural (car utilisation de plus de code pour la même chose). Plusieurs améliorations de ce dernier ont permis d'accélérer l'exécution de programme, dont voici la version la plus récente en code ouvert. Une version est en cours d'écriture avec épuration complète du code et migration vers une philosophie Fonctionnelle. Le multi paradigme de JavaScript est tout simplement génial !

Interaction entre le Scripting et une interface graphique

```

/* ** ----- ** */
/* ** --          PONG 2          -- ** */
/* ** ----- ** */
/* ** -- Auteur      :   Pierre Contri      -- ** */
/* ** -- Cree le     :   07/06/2006        -- ** */
/* ** -- Modifie le  :   24/12/2008        -- ** */
/* ** -- Update3 le  :   26/07/2015        -- ** */
/* ** ----- ** */
/* ** -- Version 1.0 :   deplacement chariot et balle      -- ** */
/* ** -- Version 1.1 :   pause plus grossissement balle   -- ** */
/* ** -- Version 1.2 :   trois balles                      -- ** */
/* ** -- Version 1.3 :   debut de gestion de la souris    -- ** */
/* ** -- Version 1.4 :   creation des briques              -- ** */
/* ** -- Version 1.5 :   gestion des briques               -- ** */
/* ** -- Version 2.0 :   code de qualite                   -- ** */
/* ** -- Version 2.1 :   passage en code objet            -- ** */
/* ** -- Version 2.2 :   simplification sur objets        -- ** */
/* ** -- Version 2.3 :   compatibilite FireFox            -- ** */
/* ** -- Version 2.4 :   simplification du code            -- ** */
/* ** -- Version 2.5 :   alignement des briques sur grille -- ** */
/* ** -- Version 2.6 :   acceleration calculs             -- ** */
/* ** -- Version 2.7 :   pouvoirs sur briques             -- ** */
/* ** -- Version 2.8 :   reduction du code                -- ** */
/* ** -- Version 2.9 :   correction interpreteur JavaScr  -- ** */
/* ** -- Version 3.0 :   amelioration qualite code        -- ** */
/* ** -- Version 3.1 :   brique avec double resistance    -- ** */
/* ** ----- ** */

// Declaration des objets du jeu
// Variables globales pour le joueur
var nbBalls = 9;
var nbBriques = 60;
var deltadepl = 3; // pour toutes les balles
var timeOutdepl = 5; // temps en millisecondes de boucle du jeu
var isIE = (window.event) ? 1 : 0; // verification du navigateur
(pour les anciens IE6 / Netscape 4)

// Variables de la classe __main__
var deplScreen = { x: 0, y: 0 };
var basculeTriche = false;
var precisionErreur = 1;
var graphismeImg = true;
var divJeu = null;
var tabBalls = null;
var tabBriques = null;
var carriage = null;

```


Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
// Redefinition de l'objet Array
// Ajout d'une methode remove
Array.prototype.remove = function (obj) {
    var tmpArray = new Array();
    for (i = 0, len = this.length; i < len; i++) {
        if (this[i] != obj) tmpArray.push(this[i]);
    }
    return tmpArray;
    /*
    for (i = 0, len = this.length; i < len; i++) {
        if (this[i] == obj) {
            return this.slice(i, 1);
        }
    }
    return null;
    */
};

// Objet "Balle"
function Balle(numero) {
    // variables
    this.element = document.createElement('DIV');
    this.element.id = "movBall" + numero;
    this.element.name = "balle";
    this.element.className = "Ball";
    this.printForm = function () {
        this.element.innerHTML = (graphismeImg)
            ? "<img src='balle.jpg' style='width: 16px;"
              + " height: 16px; border: 0px none;'/>" : "O";
    };
    this.printForm();
    // randomisation du positionnement des balles
    this.deplX = Math.floor((deplScreen.x) * Math.random());
    this.deplY = Math.floor((deplScreen.y) * Math.random());
    this.deplXPos = Math.floor(Math.random());
    this.deplYPos = 0;

    // placer la balle dans le jeu
    divJeu.appendChild(this.element);

    // methodes
    // changer la balle de sens
    this.changeBallSens = function (sens) {
        if (sens == 'X') this.deplXPos = !this.deplXPos;
        else this.deplYPos = !this.deplYPos;
    };
};
```

```
// rafraichissement de la balle (nouvelle position)
this.refresh = function () {
    this.element.style.left = this.deplX + "px";
    this.element.style.top = this.deplY + "px";
};

// tester la balle pour savoir si elle fait encore
// partie de l'air de jeu
this.isInArea = function () {
    // verifier que la balle tombe bien sur le chariot
    // et qu'elle est presente sur le terrain
    if ((this.deplY + deltadepl) >= deplScreen.y) {
        // sortir la balle du jeu ou pas
        return (this.deplX >= carriage.posCarriage
            && this.deplX <=
                (carriage.posCarriage + (carriage.getSize().x))
            || basculeTriche);
    }
    return true;
};

// effacer la balle
this.killBall = function () {
    divJeu.removeChild(this.element);
    tabBalls = tabBalls.remove(this);
    nbBalls--;
    // enlever la double palette si une balle se perd
    if (carriage.doubleCarriage) {
        carriage.doubleCarriage = false;
        carriage.printForm();
    }
};

// casser les briques sur son passage
this.breakBrique = function () {
    if (tabBriques == null) return false;
    var breakB = false;
    // parcourir les briques pour savoir
    // si la balle est dans la zone de l'une d'entre-elles
    for (idxBrique = 0, lenB = tabBriques.length;
        idxBrique < lenB; idxBrique++) {
        var tmpBrique = tabBriques[idxBrique];
        var intersect = intersectBallBrique(this, tmpBrique);
        if (intersect.breakBrique) {
            tmpBrique.breakBrique();
        }
    }
};
```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
        this.changeBallSens(intersect.sens);
        breakB = true;
    }
    return breakB;
};

// déplacement graphique de la balle
this.move = function () {
    // horizontal
    if (this.deplX + deltadepl < deplScreen.x && this.deplXPos){
        this.deplX += deltadepl;
    }
    else if (this.deplX <= 0) {
        this.deplX += deltadepl;
        this.deplXPos = 1;
    }
    else {
        this.deplX -= deltadepl;
        this.deplXPos = 0;
    }

    // vertical
    if (this.deplY + deltadepl < deplScreen.y && this.deplYPos){
        this.deplY += deltadepl;
    }
    else if (this.deplY <= 0) {
        this.deplY += deltadepl;
        this.deplYPos = 1;
    }
    else {
        this.deplY -= deltadepl;
        this.deplYPos = 0;
    }

    this.refresh();

    // si la balle n'est pas dans l'air de jeu,
    // la supprimer
    if (!this.isInArea())
        this.killBall();
    else
        this.breakBrique();
};
}
```

```

function intersectBallBrique(tmpBall, tmpBrique) {
    var intersect = { breakBrique: false, sens: 'X' };

    if (tmpBrique != null && tmpBall != null) {
        var Xball = tmpBall.deplX + Math.floor(((isIE)
            ? tmpBall.element.offsetWidth
            : tmpBall.element.clientWidth) / 2);
        var Yball = tmpBall.deplY + Math.floor(((isIE)
            ? tmpBall.element.offsetHeight
            : tmpBall.element.clientHeight) / 2);

        var X1brique = tmpBrique.element.offsetLeft;
        var X2brique = X1brique + ((isIE)
            ? tmpBrique.element.offsetWidth
            : tmpBrique.element.clientWidth);

        var Y1brique = tmpBrique.element.offsetTop;
        var Y2brique = Y1brique + ((isIE)
            ? tmpBrique.element.offsetHeight
            : tmpBrique.element.clientHeight);

        // prise en compte d'erreur de calcul processeur
        if (((X1brique <= Xball && Xball <= X2brique) &&
            (Math.abs(Yball - Y2brique) <= precisionErreur))
            ||
            ((X1brique <= Xball && Xball <= X2brique) &&
            (Math.abs(Yball - Y1brique) <= precisionErreur))) {
            intersect.breakBrique = true;
            intersect.sens = 'Y';
        }
        else if (((Y1brique <= Yball && Yball <= Y2brique) &&
            (Math.abs(Xball - X2brique) <= precisionErreur))
            ||
            ((Y1brique <= Yball && Yball <= Y2brique) &&
            (Math.abs(Xball - X1brique) <= precisionErreur))) {
            intersect.breakBrique = true;
            intersect.sens = 'X';
        }
    }

    return intersect;
}

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
// Objet Brique
function Brique(numero) {
    this.numero = numero;
    this.element = document.createElement('div');
    this.element.id = "brique" + numero;
    this.element.name = "brique";
    this.element.className = "Brique";
    this.element.resistance = 1;
    // type de brique
    // 1) multiplier les balles
    // 2) doubler le chariot
    // 3) brique double résistance
    var typeBrique = Math.floor(5 * Math.random());
    if (typeBrique == 3)
        this.element.resistance = 2;

    this.printForm = function () {
        var isBrokenBrique = (typeBrique == 3) &&
        (this.element.resistance == 1);
        this.element.innerHTML = (graphismeImg) ?
            ((!isBrokenBrique)
            ? "<img src='brique.jpg' style='width: 45px; height: 26px;'/>" :
            "<img src='brokenbrique.jpg' style='width:45px;height:26px;'/>")
            : ((!isBrokenBrique)
            ? "<table border=\"2\" \"
            + \" class=\"InterieurBrique\"><tr>\"
            + "<td style=\"border-color: blue;\">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&\"
            + "</td><td style=\"border-color: blue;\">&nbsp;&nbsp;&nbsp;&nbsp;&\"
            + "</td><td style=\"border-color: blue;\">&nbsp;&nbsp;&nbsp;&nbsp;&\"
            + "</td></tr></table>\"
            : "<table border=\"2\" class=\"InterieurBrique\"><tr>\"
            + "<td style=\"border-color: blue;\">&nbsp;&nbsp;&nbsp;&nbsp;&\"
            + "&nbsp;&nbsp;&nbsp;&\"</td><td style=\"border-color: blue;\">&nbsp;&nbsp;&\"
            + "&nbsp;&nbsp;&\"</td></tr></table>");
    };
    this.printForm();
    // ajout de la brique au jeu
    divJeu.appendChild(this.element);

    this.briqueSize = {
        x: (isIE) ? this.element.offsetWidth
            : this.element.clientWidth,
        y: (isIE) ? this.element.offsetHeight
            : this.element.clientHeight
    };
};
```

```

this.getRandomPosition = function () {
    // positionnement aleatoire sur la grille
    var tmpPos = {
        x: (deplScreen.x) * Math.random(),
        y: (deplScreen.y * 3 / 5) * Math.random()
    };

    // alignement sur une grille virtuelle
    tmpPos.x = Math.floor(tmpPos.x / this.briqueSize.x)
        * this.briqueSize.x;
    tmpPos.y = Math.floor(tmpPos.y / this.briqueSize.y)
        * this.briqueSize.y;

    return tmpPos;
};

this.isEqualPosition = function (tmpBrique) {
    return (this.posRnd.x == tmpBrique.posRnd.x
        && this.posRnd.y == tmpBrique.posRnd.y);
};

// positionner la brique sur le jeu à un endroit libre
do {
    this.posRnd = this.getRandomPosition();
} while (containtBriquePosition(this));

this.element.style.left = this.posRnd.x + "px";
this.element.style.top = this.posRnd.y + "px";

// destruction de la brique
this.breakBrique = function () {
    // performances problem
    //this.element.resistance--;
    switch (typeBrique) {
        case 0:
            var nbBallDem = 3;
            //if (nbBalls < nbBallDem) nbBallDem = nbBalls;
            nbBallDem = nbBallDem - tabBalls.length;
            for (i = 0; i < nbBallDem; i++)
                tabBalls.push(new Balle(tabBalls.length));
            break;
        case 1:
            carriage.doubleCarriage = true;
            carriage.printForm();
            break;
    }
}

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
        case 3:
            // double resistance
            // performances problem
            // must get the 'this.element.resistance'
            // decrease here
            this.element.resistance--;
            if (this.element.resistance > 0) {
                this.printForm();
                return false;
            }
            break;
        default:
            break;
    }
    divJeu.removeChild(this.element);
    tabBriques = tabBriques.remove(this);
    return true;
};
}

function containtBriquePosition(searchBrique) {
    for (i = 0, tabBriquesLength = tabBriques.length;
        i < tabBriquesLength; i++) {
        if (searchBrique.isEqualPosition(tabBriques[i]))
            return true;
    }
    return false;
}

// Objet chariot
function Carriage() {
    this.element = document.createElement('DIV');
    this.element.id = "carriage";
    this.element.className = "Carriage";
    var tmpX = 20; // taille temporaire du chariot

    this.printForm = function () {
        this.element.innerHTML = (graphismeImg
            ? "<img src='palette.jpg'/>" : "_____");
        if (this.doubleCarriage)
            this.element.innerHTML += this.element.innerHTML;
        tmpX = this.getSize().x;
    };
};
```

```
this.getSize = function () {
    return {
        x: (isIE) ? this.element.offsetWidth
          : this.element.clientWidth,
        y: (isIE) ? this.element.offsetHeight
          : this.element.clientHeight
    };
};

this.printForm();

this.deltaCarriage = 20;
this.element.style.top = deplScreen.y;
this.posCarriage = (deplScreen.x / 2);
this.element.style.left = this.posCarriage + "px";
this.doubleCarriage = false;

this.refresh = function () {
    if (!basculeTriche)
        this.element.style.left = this.posCarriage + "px";
};

this.move = function (newPosition) {
    if ((newPosition - (tmpX / 2)) > 0
        && (newPosition + (tmpX / 2)) <= deplScreen.x
        && !basculeTriche) {
        this.posCarriage = newPosition - (tmpX / 2);
        this.refresh();
    }
};

this.moveLeft = function () {
    var tmpPosL = this.posCarriage + (tmpX / 2)
        - this.deltaCarriage;
    this.move(tmpPosL);
};

this.moveRight = function () {
    var tmpPosR = this.posCarriage + (tmpX / 2)
        + this.deltaCarriage;
    this.move(tmpPosR);
};
```


Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
this.triche = function () {
    basculeTriche = !basculeTriche;
    if (basculeTriche) {
        this.element.innerHTML =
"
        this.element.style.left = "0px";
    }
    else {
        this.printForm();
    }
};

// ajouter le chariot au jeu
divJeu.appendChild(this.element);
// recuperer la taille du chariot
tmpX = this.getSize().x;
}

function Init() {
    deplScreen = getSizeScreen();
    divJeu = document.getElementById("jeu");
    if (!divJeu) return false;

    // Creer le chariot
    carriage = new Carriage();

    // Creation d'une balle
    tabBalls = new Array();
    tabBalls.push(new Balle(0));

    // Initialisation des briques
    tabBriques = new Array();
    for (i = 0; i < nbBriques; i++)
        tabBriques.push(new Brique(i));

    // gestion clavier et souris
    document.onkeypress = handlerKey;
    document.onkeydown = movCarriageByKeyboard;
    document.onmousemove = movCarriageByMouse;

    // lancement du jeu
    setTimeout('goBall()', timeOutdepl);
    return true;
}
```

```
function getSizeScreen() {  
    return {  
        x: ((isIE) ? document.body.offsetWidth  
            : document.body.clientWidth) - 40,  
        y: ((isIE) ? document.body.offsetHeight  
            : document.body.clientHeight) - 30  
    };  
}
```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
function goBall() {
    // bouger les balles
    for (i = 0, tabBallsLen = tabBalls.length;
        i < tabBallsLen; i++) {
        tabBalls[i].move();
    } // end for

    // perdu si plus de balles
    if (tabBalls == null || tabBalls.length == 0) {
        if (confirm("Game Over !\nStart a new part ?"))
            document.location.reload();
        else
            return false;
    }

    // relancer le jeu si plus de briques
    if (tabBriques == null || tabBriques.length == 0) {
        if (confirm("Congratulations !\nStart a new part ?"))
            document.location.reload();
        else
            return false;
    }

    setTimeout('goBall()', timeOutdepl);
}

function handlerKey(e) {
    var keyPress = (window.event) ? event.keyCode : e.which;

    if (keyPress == 43 && timeOutdepl > 1) timeOutdepl--;
    else if (keyPress == 45) timeOutdepl++;
    else if (keyPress == 42) deltadepl++;
    else if (keyPress == 47 && deltadepl > 1) deltadepl--;
    else if (keyPress == 48) {
        for (i = 0, tabBallsLen = tabBalls.length;
            i < tabBallsLen; i++) {
            if (tabBalls[i].element.innerHTML == "o")
                tabBalls[i].element.innerHTML = "O";
            else
                tabBalls[i].element.innerHTML = "o";
        }
    }
    else if (keyPress >= 49 && keyPress <= 57) { // de 1 a 9
        // nombre de balles tappees au clavier
        var nbBallDem = String.fromCharCode(keyPress);
        if (nbBalls < nbBallDem) nbBallDem = nbBalls;
    }
}
```

Interaction entre le Scripting et une interface graphique

```

        nbBallDem = nbBallDem - tabBalls.length;
        for (i = 0; i < nbBallDem; i++)
            tabBalls.push(new Balle(tabBalls.length));
    }
    else if (keyPress == 32) carriage.triche(); // espace
    else if (keyPress == 27) // escape
        alert("Pause, v'la le chef !\nOK pour continuer ...");
    else if (keyPress == 66) { // 'B'
        graphismeImg = !graphismeImg;
        // changer le graphisme des balles
        if (tabBalls == null) return false;
        for (i = 0, tabBallsLen = tabBalls.length;
            i < tabBallsLen; i++)
            tabBalls[i].printForm();
        // changer le graphisme des briques
        if (tabBriques == null) return false;
        for (i = 0, tabBriquesLen = tabBriques.length;
            i < tabBriquesLen; i++)
            tabBriques[i].printForm();
        // changer le graphisme du chariot
        carriage.printForm();
    }
    else if (keyPress == 68) { // 'D'
        carriage.doubleCarriage = !carriage.doubleCarriage;
        carriage.printForm();
    }
    else if (keyPress >= 65) { // a partir de 'A'
        for (i = 0, tabBallsLen = tabBalls.length;
            i < tabBallsLen; i++)
            tabBalls[i].element.innerHTML =
                String.fromCharCode(keyPress);
    }
    return true;
}

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
function movCarriageByKeyboard(e) {
    switch ((window.event) ? event.keyCode : e.which) {
        case 37: //deplacement vers la gauche
            carriage.moveLeft();
            break;
        case 39: //deplacement vers la droite
            carriage.moveRight();
            break;
        case 38: //acceleration du deplacement chariot
            carriage.deltaCarriage++;
            break;
        case 40: // decelleration du deplacement chariot
            if (carriage.deltaCarriage > 1)
                carriage.deltaCarriage--;
            break;
        default: // ne rien faire
            break;
    }
}

function movCarriageByMouse(e) {
    carriage.move((window.event) ? event.x : e.clientX);
}
```

Sans le programme principal (en HTML), le code présenté n'aurait pas d'intérêt.
Voici le plateau du jeu.

Interaction entre le Scripting et une interface graphique

```

<!-- ----- -->
<!-- --          PONG 2          -- -->
<!-- ----- -->
<!-- -- Auteur      : Pierre Contri      -- -->
<!-- -- Cree le     : 07/06/2006        -- -->
<!-- -- Modifie le  : 08/06/2006        -- -->
<!-- -- Version     : 1.0 deplacement chariot et balle -- -->
<!-- -- Version     : 1.1 pause plus grossissement balle -- -->
<!-- -- Version     : 1.2 trois balles   -- -->
<!-- -- Version     : 1.3 debut de gestion de la souris -- -->
<!-- -- Version     : 1.4 creation des briques -- -->
<!-- -- Version     : 1.5 gestion des briques -- -->
<!-- -- Version     : 2.0 code de qualite -- -->
<!-- -- Version     : 2.1 passage du code js en objet -- -->
<!-- -- Version     : 2.2 compatibilite avec FireFox -- -->
<!-- ----- -->
<html>
  <head>
    <title>Pong 2 - The return(true);</title>
    <link href="/pong2.css" rel="stylesheet" type="text/css">
    <script language="javascript" type="text/javascript"
src="/pong2obj.secure.js"></script>
  </head>
  <body id="jeu" onload="javascript:Init();">
    <!-- Les balles -->
    <!-- Creation des balles automatiques -->
    <!-- Les briques -->
    <!-- Creation des briques automatique -->
    <!-- Le chariot -->
    <!-- Creation du chariot automatique -->
  </body>
</html>

```

Nous nous apercevons que le plateau du jeu ne comporte aucune ligne utile (entre les balises "body"), et que seule la ligne d'initialisation est présente. Tout est fait en JavaScript.

Pour ceux qui aimeraient critiquer la qualité du code, je les invite à venir coder avec moi avec des outils de base, tels que Leafpad ou Notepad.; ceci à fin d'améliorer le code tout en maintenant une compatibilité avec les anciens systèmes (Netscape 4).

PowerShell

Contenu de ce chapitre

| | |
|--|-----|
| PowerShell..... | 167 |
| Pourquoi utiliser ce langage..... | 168 |
| Afficher simplement 'Hello World'..... | 169 |
| Comment utiliser les scripts par la console..... | 170 |
| Aller plus loin avec le Scripting en mode console..... | 171 |
| Travailler avec Azure et la partie System..... | 173 |
| Interaction entre le Scripting et une interface graphique..... | 183 |

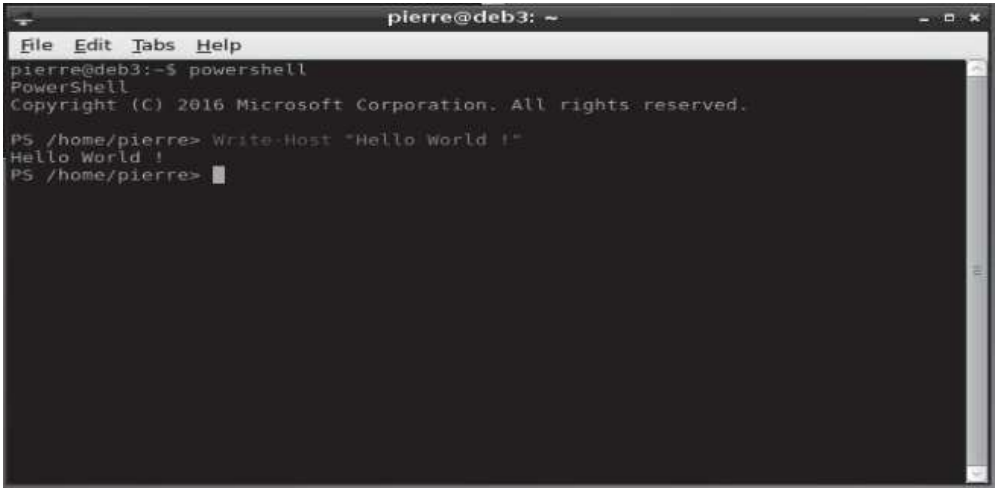
Pourquoi utiliser ce langage

Tout simplement génial, ce langage de script intègre tout le Framework DotNet. Cela sous-entend, que l'on peut écrire des cmdlet (nom donné aux instructions PowerShell), mais aussi appeler directement un objet ou un code compilé en DotNet. Simple et rapide, car interprété dans sa console, nous pouvons réaliser un test de fonction .Net à la place de créer un projet console, d'écrire un cas de test, de compiler et enfin de récupérer le traitement. PowerShell se trouve aussi bien sous Windows que sous Linux.

Afficher simplement 'Hello World'

Lancer simplement la console PowerShell (Windows et Linux).

```
Write-Host "Hello World !"
```

A screenshot of a terminal window titled 'pierre@deb3: ~'. The window has a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The terminal shows the user running 'powershell' at the prompt 'pierre@deb3:~\$'. This opens a PowerShell prompt 'PS /home/pierre>'. The user enters 'Write-Host "Hello World !"', which outputs 'Hello World !' on the next line. The prompt returns to 'PS /home/pierre>'.

```
pierre@deb3: ~  
File Edit Tabs Help  
pierre@deb3:~$ powershell  
PowerShell  
Copyright (C) 2016 Microsoft Corporation. All rights reserved.  
  
PS /home/pierre> Write-Host "Hello World !"  
Hello World !  
PS /home/pierre> █
```

Illustration 24: Hello World en PowerShell sous Debian

Comment utiliser les scripts par la console

Deux types principaux de scripts PowerShell sont facilement utilisables. Les modules PowerShell (psm1) et les scripts à proprement parler (ps1).

En exécutant un module dans la console PowerShell, nous pouvons utiliser les fonctions et objets chargés précédemment.

Nous avons aussi un interpréteur de commande 'ISE' permettant d'écrire du code dans un fichier et de l'interpréter directement avec des options de débogage.

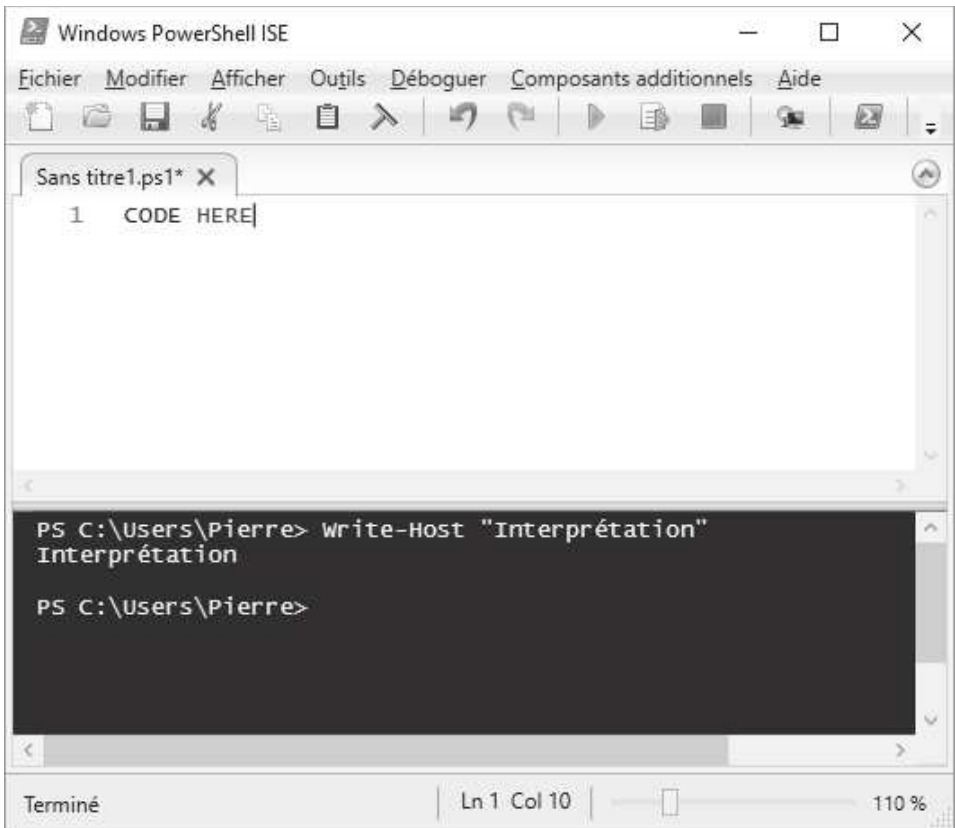


Illustration 25: ISE -> Editeur PowerShell intégré à Windows

Aller plus loin avec le Scripting en mode console

Dans un premier temps, nous allons créer un programme très court permettant de minimiser le script du pong2 présenté au chapitre JavaScript.

```
# first step: try to remove all in one => not good
#cat pong2obj.js | Out-String | foreach-object
{[Regex]::Replace([Regex]::Replace($_,'(\\/.*)' |
(\\/.*.\\*\\/),''),'')'([\\t\\n\\r ]+)',' ')} | Out-File pong2obj.secure.js

# second step: remove comments and after remove carriage return with
an intermediate file (temporary)
#cat pong2obj.js | Out-String | foreach-object
{[Regex]::Replace($_,'(\\/\\.*)*','')} | Out-File
pong2obj.secure.js.tmp
#cat pong2obj.secure.js.tmp | Out-String | foreach-object
{[Regex]::Replace($_,'([\\t\\r\\n ]+)',' ')} | Out-File
pong2obj.secure.js
#remove-item pong2obj.secure.js.tmp

# third step: remove comments and after remove carriage return but
into pipeline and not temporary file
$strContent = Get-Content ./pong2obj.js
# get the header of the script
$header = $strContent | Select-Object -First 27
# get content and remove all comments
#$strContent | Select-Object $_
#Write-Verbose -Message "line" -Verbose
```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
$strContent = $strContent `
| Out-String `
| ForEach-Object {[Regex]::Replace($_, '(//.*)*', '')} `
| Out-String `
| ForEach-Object {[Regex]::Replace($_, '(\\\/*.*\\\/)*', '')} `
| Out-String `
| ForEach-Object {[Regex]::Replace($_, '([\n\r]+)', '')} `
| Out-String `
| ForEach-Object {[Regex]::Replace($_, '[\t ]+', ' ')} `
| Out-String `
| ForEach-Object {[Regex]::Replace($_, '(\W) ( )', '$1')} `
| Out-String `
| ForEach-Object {[Regex]::Replace($_, '( ) (\W)', '$2')}

$strHeader + $strContent | Out-File pong2obj.secure.js
```

Dans cet exemple, nous utilisons des Cmdlet tout à fait standard, mais nous faisons également référence au Framework DotNet directement, par l'utilisation des Regex (System.Regex) pour traiter les chaînes de caractères extraites dans le traitement.

Travailler avec Azure et la partie System

Cette sous partie va démontrer l'importance de travailler avec PowerShell.

Plaçons nous dans un exemple d'entreprise dynamique ayant besoin de créer des pools de machines virtuelles dans l'Azure.

La première étape, pour un ingénieur Système est de créer son premier pool de machines virtuelles à la main, puis, de le mettre à disposition pour les utilisateurs/développeurs.

Au regard de certains choix "business", la compagnie décide de faire supprimer ce pool, puis redemande aux ingénieurs systèmes de créer une nouvelle ferme pour ses développements expérimentaux, puis ses tests.

Par la suite, le projet ayant réussi, il faut recréer deux fermes pour l'intégration, ainsi que le système de mise en production.

Reproduisons ce schéma un certain nombre de fois en relation avec les décisions stratégiques de la société.

Nous nous apercevons alors que les ingénieurs systèmes ont beaucoup de clic, beaucoup d'attente et beaucoup de configurations pour les différents projets, qui pour eux reviennent à un travail répétitif, et donc inintéressant.

Grâce à PowerShell, il est possible de scripter la création d'une ferme de serveurs et de répéter l'installation de cette ferme avec des paramètres de configuration définis au lancement de chaque installation (donc en script).

Pour débiter cet exercice, nous allons créer un Module PowerShell (AzureManagement.psm1) contenant des fonctions de base qui seront réutilisées dans nos scripts ultérieurs.

```
<# ##### Create Cloud Services ##### #>

function createAzureService($serviceName, $serviceLocation)
{
    Write-Host "$(Get-Date -f $timestampFormat) - Cloud service
creation - Started " -foregroundcolor "red"

    # check for existence
    $service = Get-AzureService -ServiceName $serviceName
-ErrorVariable errPrimaryService -Verbose:$false -ErrorAction
"SilentlyContinue"

    if ($service -ne $null)
    {
        Write-Host "$(Get-Date -f $timestampFormat) - Error occurred
- $($serviceName) Service already exists " -foregroundcolor "red"
        return
    }

    # create new service
    New-AzureService -ServiceName $serviceName -Location
$serviceLocation -ErrorVariable errPrimaryService -Verbose:$false
-ErrorAction "SilentlyContinue" | Out-Null

    if ($errPrimaryService[0] -ne $null)
    {
        Write-Host "$(Get-Date -f $timestampFormat) - Error occurred
- $($serviceName) Service failed - $errPrimaryService[0] "
-foregroundcolor "red"
        return
    }

    Write-Host "$(Get-Date -f $timestampFormat) - Cloud service
creation - Completed " -foregroundcolor "red"
}
```

```
<# ##### Create Blob Container in Storage ##### #>

function createBlobContainer($containerName)
{
    Write-Host "$(Get-Date -f $timestampFormat) "
        + " - Storage container creation - Started "
        -foregroundcolor "red"

    $blobContainer = Get-AzureStorageContainer -Name $containerName
        -ErrorVariable errContainer -Verbose:$false
        -ErrorAction "SilentlyContinue"

    if($blobContainer -eq $null)
    {
        New-AzureStorageContainer -Name $containerName
            -Verbose:$false | Out-Null
    }

    Write-Host "$(Get-Date -f $timestampFormat) "
        + "- Storage container creation - Completed "
        -foregroundcolor "red"
}
```


Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
<# ##### ProductionDeployment to Service ##### #>

function createProductionDeployment($packageLocation,
$configLocation, $serviceName)
{
    Write-Host "$(Get-Date -f $timestampFormat) "
        + "- Cloud service deployment - Started "
        -foregroundcolor "red"

    # check for deployment existence
    $serviceDeployment = (Get-AzureDeployment
                        -ServiceName $serviceName
                        -Slot Production
                        -ErrorAction "SilentlyContinue"
                        -Verbose:$false)

    if ($serviceDeployment -ne $null)
    {
        Write-Host "$(Get-Date -f $timestampFormat) - Error occurred
- $($serviceName) deployment already exists " -foregroundcolor "red"
        return
    }

    # create new deployment
    New-AzureDeployment -Slot "Production"
        -Package $packageLocation -Configuration $configLocation
        -label $serviceName -ServiceName $serviceName
        -Name $serviceName -ErrorVariable errPrimaryDeployment
        -Verbose:$false | Out-Null

    if ($errPrimaryDeployment[0] -ne $null)
    {
        Write-Host "$(Get-Date -f $timestampFormat) - Error"
            + " occurred - Creating new deployment in "
            + " $($serviceName) service"
            -foregroundcolor "red"

        return
    }

    Write-Host "$(Get-Date -f $timestampFormat) - Cloud service "
        + "deployment - Completed "
        -foregroundcolor "red"
}
```

```
<# ##### Check health of Cloud Service Deployment ##### #>

function checkCloudServiceHealth($serviceName)
{
    Write-Host "$(Get-Date -f $TimeStampFormat) - Cloud service"
        + " deployment - Creating VM "
        -foregroundcolor "red"

    $serviceHealth = (Get-AzureDeployment -ServiceName $serviceName
-Slot Production -Verbose:$false)

    While($serviceHealth.RoleInstanceList[0].InstanceStatus
        -ne "ReadyRole")
    {
        Write-Host "." -NoNewLine -foregroundcolor "yellow"
        Start-Sleep -s 30
        $serviceHealth = (Get-AzureDeployment
                        -ServiceName $serviceName
                        -Slot Production -Verbose:$false)

        if($serviceHealth.RoleInstanceList[0].InstanceStatus
            -eq "ReadyRole")
        {
            Write-Host "." -foregroundcolor "yellow"
        }
    }

    Write-Host "$(Get-Date -f $TimeStampFormat) - Cloud service"
        + " deployment - Up and running " -foregroundcolor "red"
}
```

Analysez (vos besoins), Etudiez, Scriptez L'art d'utiliser du scripting pour un projet

Une fois ce module défini, créons maintenant une ferme de serveurs

```
Import-Module azure

# Connect to the Azure Account
#Add-AzureAccount
Get-AzureSubscription -Current

# select the "Windows Server 2012 R2 Datacenter" on the catalog list
$ImageName = (Get-AzureVMImage
| Where { $_.ImageFamily -eq "Windows Server 2012 R2 Datacenter" }
| Sort PublishedDate -Descending
| Select-Object -First 1).ImageName

# define the parameters for all subnet machines
# Caution: If serviceName does not exists, do not forget to create
it with CreateAzureAZAvailabilityGrp_Ressources.ps1 script !
$serviceName      = "AZPOCAvailabilityGrp"
$loginAdmin        = "admin"
$pwdAdmin           = "admin"
$vnnetName          = "Azure-VNet01"
$subnetName         = "Subnet-1"
$azureVNet          = Get-AzureVNetSite | ? { $_.Name -like $vnnetName }

$vmconfigList      = @()
$vmconfigList += @{ vmName="AZAGSV06"; vmIP = "192.168.202.45" ;
vmSize = "ExtraSmall" } # Farm Server
$vmconfigList += @{ vmName="AZAGSV07"; vmIP = "192.168.202.46" ;
vmSize = "ExtraSmall" } # IIS primary server
$vmconfigList += @{ vmName="AZAGSV08"; vmIP = "192.168.202.47" ;
vmSize = "ExtraSmall" } # IIS secondary server
$vmconfigList += @{ vmName="AZAGSV09"; vmIP = "192.168.202.48" ;
vmSize = "ExtraSmall" } # IIS secondary server

# Get the Azure Storage account "storage01"
$azstorage = Get-AzureStorageKey
              -StorageAccountName "aztstenvironmentstorage"

# Get certificate and subscription id
$cert = Get-AzureAccount
        | Where-Object { $_.Type -like "Certificate" }
$subid = $cert.Subscriptions
```

```

$vmList = @()

function CreateVM {
    Param (
        [Parameter(Mandatory=$True)]
        [String]$vmName,
        [Parameter(Mandatory=$True)]
        [String]$vmIP,
        [String]$vmSize = "Medium"
    )

    $vmTmp = New-AzureVMConfig -Name $vmName
        -InstanceSize $vmSize -ImageName $ImageName
        | Add-AzureProvisioningConfig -Windows
        -AdminUsername $loginAdmin -Password $pwdAdmin
        | Set-AzureSubnet -SubnetNames $subnetName
        | Set-AzureStaticVNetIP -IPAddress $vmIP

    return $vmTmp
}

$vmconfigList | ForEach-Object {
    $vmList += ,$(CreateVM -vmName $_["vmName"] -vmIP $_["vmIP"]
        -vmSize $_["vmSize"])
}

# Show the collection
$vmList | Format-List

# Create the VMs
New-AzureVM -ServiceName $serviceName -VMs $vmList -VNetName
$vnetsName -WaitForBoot

```

Analysez (vos besoins), Etudiez, Scriptez L'art d'utiliser du scripting pour un projet

Maintenant que Le DevOps a son script magique pour générer son environnement de serveurs complet, retournons côté développeur, avec un script de création d'une machine SharePoint dans le Cloud.

```
Import-Module Azure

# Connect to the Azure Account
Add-AzureAccount

Split-Path -parent $MyInvocation.MyCommand.Definition

Import-Module "..\..\Modules\AzureManagement.psml"

$serviceName      = "SPDevGrp"
$location          = "West Europe"
$affinityGrpName   = $serviceName

# Resources Group
break
CreateAzureService -ServiceName $serviceName -serviceLocation
$location
$resourcesGrp = get-azureservice
                | where-object { $_.ServiceName -like $serviceName }
                # | Remove-AzureService

# Get or create affinity group
if(@(Get-AzureAffinityGroup
    | ? Name -eq $affinityGrpName).Count -eq 0) {
    New-AzureAffinityGroup -Name $affinityGrpName -Location $location
}
$affinityGrp = Get-AzureAffinityGroup -Name $affinityGrpName

# select the "Windows Server 2012 R2 Datacenter" on the catalog list
$imageWin = (Get-AzureVMImage
    | Where { $_.ImageFamily -eq
        "Windows Server 2012 R2 Datacenter" }
    | Sort PublishedDate -Descending
    | Select-Object -First 1).ImageName

$imageSQL = (Get-AzureVMImage
    | Where { $_.ImageFamily -eq
        "SQL Server 2014 SP1 Standard on Windows Server 2012 R2" }
    | Sort PublishedDate -Descending
    | Select-Object -First 1).ImageName
```

```
# not available
$ImageSP = (Get-AzureVMImage
            | Where-Object ImageFamily -like "*SharePoint*2013*"
            | Sort PublishedDate -Descending).ImageName

# Create an internal subnet

# create a very hight power machine
$loginAdmin      = "administrator"
$pwdAdmin        = "P@ssw0rd$"
$azstorageName   = "storespdev"
$resourcesGrpName = $resourcesGrp.ServiceName

$vmconfigList = @()
$vmconfigList += @{ vmName="SPDev02"; vmSize = "Standard_D12_v2";
vmImg = $ImageWin } # Db Dev

# Get or add the Azure Storage account "devstorage"
if(@(Get-AzureStorageAccount
    | ? Name -like $azstorageName).Count -eq 0) {
    New-AzureStorageAccount -StorageAccountName $azstorageName
        -AffinityGroup $affinityGrpName -Type "Standard_LRS"
        # -Location $location
}
$azstorage = Get-AzureStorageKey -StorageAccountName $azstorageName

# Get certificate and subscription id
$cert = Get-AzureAccount
    | Where-Object { $_.Type -like "Certificate" }
$subid = $cert.Subscriptions

$vmList = @()

function CreateVM {
    Param (
        [Parameter(Mandatory=$True)]
        [String]$vmName,
        #[Parameter(Mandatory=$True)]
        #[String]$vmIP,
        [String]$vmSize = "Medium",
        $vmImg
    )
}
```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
$vmTmp = New-AzureVMConfig -Name $vmName -InstanceSize $vmSize
    -ImageName $vmImg -MediaLocation $resourcesGrp.Url
    |
    Add-AzureProvisioningConfig -Windows
        -AdminUsername $loginAdmin -Password $pwdAdmin
    # |
    #Set-AzureSubnet -SubnetNames $subnetName |
    #Set-AzureStaticVNetIP -IPAddress $vmIP

    return $vmTmp
}

$vmconfigList | ForEach-Object {
    if($_["vmImg"] -eq $null) {$_["vmImg"] = $ImageName}

    $vmList += ,(CreateVM -vmName $_["vmName"] -vmSize $_["vmSize"]
        -vmImg $_["vmImg"])
}

# Show the collection
$vmList | Format-List

# Create the VMs
New-AzureVM -ServiceName $serviceName -AffinityGroup $affinityGrp
-VMs $vmList -WaitForBoot
```

Interaction entre le Scripting et une interface graphique

Grâce à l'intégration de Window Presentation Foundation dans le Framework, PowerShell bénéficie des fonctionnalités XAML définies par les interfaces WPF de DotNet.

Prenons l'exemple très connu d'informations sur le système hôte.

Nous avons une interface graphique (définie dans un fichier `main_window.xaml`), et le code PowerShell (défini dans un fichier `computer_infos.ps1`).

Le fichier `"computer_info.ps1"` est le point d'entrée du programme.

Dans ce dernier, nous commencerons par charger le composant graphique `"presentationframework"`.

Nous récupérerons ensuite le contenu du second fichier et le parcourrons en XML.

Pour la suite, le programme principal se chargera de faire le mapping entre les champs texte de l'interface et le contenu à afficher.

Analysez (vos besoins), Etudiez, Scriptez L'art d'utiliser du scripting pour un projet

Voici la partie de définition graphique en XAML

```
<Window
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="OS Details" Height="236" Width="525"
WindowStartupLocation="CenterScreen" WindowStyle='None'
ResizeMode='NoResize'>
    <Grid Margin="0,0,-0.2,0.2">
        <TextBox HorizontalAlignment="Center" Height="23"
TextWrapping="Wrap" Text="Operating System Details"
VerticalAlignment="Top" Width="525" Margin="0,-1,-0.2,0"
TextAlignment="Center" Foreground="White" Background="#FF98D6EB"/>
        <Label Content="Hostname" HorizontalAlignment="Left"
Margin="0,27,0,0" VerticalAlignment="Top" Height="30" Width="170"
Background="#FF98D6EB" Foreground="White"/>
        <Label Content="Operating System Name"
HorizontalAlignment="Left" Margin="0,62,0,0" VerticalAlignment="Top"
Height="30" Width="170" Background="#FF98D6EB" Foreground="White"/>
        <Label Content="OS Architecture" HorizontalAlignment="Left"
Margin="0,97,0,0" VerticalAlignment="Top" Height="30" Width="170"
Background="#FF98D6EB" Foreground="White"/>
        <Label Content="Available Memory" HorizontalAlignment="Left"
Margin="0,132,0,0" VerticalAlignment="Top" Height="30" Width="170"
Background="#FF98D6EB" Foreground="White"/>
        <Label Content="System Drive" HorizontalAlignment="Left"
Margin="0,167,0,0" VerticalAlignment="Top" Height="30" Width="170"
Background="#FF98D6EB" Foreground="White"/>
        <Button Name="btnExit" Content="Exit"
HorizontalAlignment="Left" Margin="0,202,0,0"
VerticalAlignment="Top" Width="525" Height="34"
BorderThickness="0"/>
        <TextBox Name="txtHostName" HorizontalAlignment="Left"
Height="30" Margin="175,27,0,0" TextWrapping="Wrap" Text=""
VerticalAlignment="Top" Width="343" IsEnabled="False"/>
        <TextBox Name="txtOSName" HorizontalAlignment="Left"
Height="30" Margin="175,62,0,0" TextWrapping="Wrap" Text=""
VerticalAlignment="Top" Width="343" IsEnabled="False"/>
        <TextBox Name="txtOSArchitecture" HorizontalAlignment="Left"
Height="30" Margin="175,97,0,0" TextWrapping="Wrap" Text=""
VerticalAlignment="Top" Width="343" IsEnabled="False"/>
        <TextBox Name="txtAvailableMemory"
HorizontalAlignment="Left" Height="30" Margin="175,132,0,0"
TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="343"
IsEnabled="False"/>
```

Interaction entre le Scripting et une interface graphique

```
<TextBox Name="txtSystemDrive" HorizontalAlignment="Left"
Height="30" Margin="175,167,0,0" TextWrapping="Wrap" Text=""
VerticalAlignment="Top" Width="343" IsEnabled="False"/>
</Grid>
</Window>
```

Et voici la partie PowerShell à proprement parlé

```
#=====
# XAML Code - Imported from Visual Studio Express WPF Application
#=====
[void]
[System.Reflection.Assembly]::LoadWithPartialName('presentationframe
work')
$xml]$XAML = Get-Content ".\main_window.xaml"

#Read XAML
$reader=(New-Object System.Xml.XmlNodeReader $xml)
try{$Form=[Windows.Markup.XamlReader]::Load( $reader )}
catch{Write-Host "Unable to load Windows.Markup.XamlReader. Some
possible causes for this problem include: .NET Framework is missing
PowerShell must be launched with PowerShell -sta, invalid XAML code
was encountered."; exit}

#=====
# Store Form Objects In PowerShell
#=====
$xml.SelectNodes("//*[@Name]") | %{Set-Variable -Name ($_.Name)
-Value $Form.FindName($_.Name)}

#=====
# Add events to Form Objects
#=====
$btnExit.Add_Click({$form.Close()})

#=====
# Stores WMI values in WMI Object from Win32_Operating System Class
#=====
$wmiOS = Get-WmiObject win32_OperatingSystem

#=====
# Extract and display informations
#=====
$aOSName = $wmiOS.name.Split("|")
$sAvailableMemory = [math]::round($wmiOS.freephysicalmemory/1000,0)
$sAvailableMemory = "$sAvailableMemory MB"
```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
#=====
# Trace in command ligne the informations
#=====
Write-Host $("Computer Name      : " + $oWMIOS.PSComputerName)
Write-Host $("OS                  : " + $aOSName[0])
Write-Host $("OS Architecture    : " + $oWMIOS.OSArchitecture)
Write-Host $("Available memory  : " + $sAvailableMemory)
Write-Host $("System Drive       : " + $oWMIOS.SystemDrive)

#=====
# Links WMI Object Values to XAML Form Fields
#=====
$txtHostName.Text = $oWMIOS.PSComputerName

#Displays OS name
$txtOSName.Text = $aOSName[0]

#Displays Available memory
$txtAvailableMemory.Text = $sAvailableMemory

#Displays OS Architecture
$txtOSArchitecture.Text = $oWMIOS.OSArchitecture

#Displays System Drive
$txtSystemDrive.Text = $oWMIOS.SystemDrive

#=====
# Shows the form
#=====
$Form.ShowDialog() | out-null
```

Le résultat est de très bonne qualité et nous aurions de la peine à imaginer que cela a été fait en PowerShell.

| Operating System Details | |
|--------------------------|------------------------------------|
| Hostname | OTTER |
| Operating System Name | Microsoft Windows 10 Professionnel |
| OS Architecture | 64 bits |
| Available Memory | 6669 MB |
| System Drive | C: |
| Exit | |

Illustration 26: Interface graphique avec PowerShell

Perl

Contenu de ce chapitre

| | |
|--|-----|
| Perl..... | 189 |
| Pourquoi utiliser ce langage..... | 190 |
| Afficher simplement 'Hello World'..... | 191 |
| Comment utiliser les scripts par la console..... | 192 |
| Aller plus loin avec le Scripting en mode console..... | 193 |
| Interaction entre le Scripting et une interface graphique..... | 195 |

Pourquoi utiliser ce langage

Perl est un magnifique langage qui a une trentaine d'années maintenant. Ce langage est rapide et avait pour but de traiter des chaînes de caractères ainsi que des fichiers de manière extrêmement rapide. Ce dernier, conçu pour une programmation Procédurale a été adapté à une écriture en Orienté Objet, mais présente de nombreuses failles au niveau de l'écriture des pointeurs; ce qui est logique car il n'était pas écrit pour cela au départ. Malheureusement, ce langage est pour moi d'un autre temps. Pour y avoir pu travailler en entreprise avec ce dernier sur des séquences de traitement batch, je n'ai trouvé aucun point positif par rapport à Python ou PHP. Perl était installé par défaut sur tous les systèmes Linux, mais, depuis 2015, nous constatons que certaines distributions ne l'intègrent plus par défaut dans l'installation. Python ayant pris sa place, c'est maintenant ce dernier qui est installé par défaut. A l'heure actuelle, en tant que chef de projet, je n'emploierai ce langage que pour des scripts simples avec des programmeurs Unix orientés Awk ou C ; utilisant une écriture procédurale. Le plus important est quand même l'utilisation d'expressions régulières pour ce langage. C'est vraiment son point fort.

Afficher simplement 'Hello World'

```
#!/usr/bin/perl
use strict;
use warnings;
print "Hello World !\n";
```

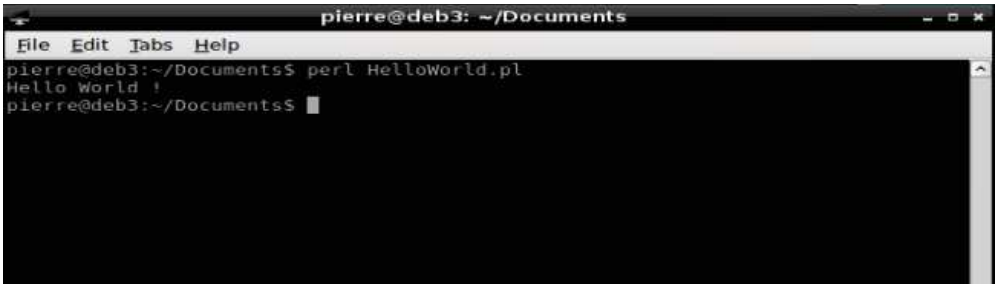


Illustration 27: Hello World en console pour Perl

Comment utiliser les scripts par la console

Avec Perl, comme beaucoup de langages de script, nous créons un fichier avec l'extension ".pl", puis nous appelons le programme créé par "perl mytestfile.pl". Il existe de nombreux IDE permettant de créer et déboguer un programme à la volée, mais je vous laisserai vous renseigner pour ces derniers, étant donné que je préfère utiliser le strict minimum d'outils externe pour ne pas me compromettre en cas de changement de système ou d'interventions "critiques" en entreprise sur les serveurs de production directement.

Un petit exemple de jeu avec les expressions régulières, le grand point fort de Perl.

```
# Un exemple de programme en Perl
$message = "À l'endroit : 'camel'.\n";
print $message;
$message =~ s/endroit/envers/;
$message =~ s/('\w+)/reverse($1)/e;
print $message;
@teams = ('cubs', 'reds', 'yankees', 'dodgers');
print "@teams\n";
exit 0
```

Aller plus loin avec le Scripting en mode console

Le Scripting en Perl est à privilégier avec une pensée orientée procédurale. Je vais donc montrer un petit script d'exemple permettant de jouer avec des pointeurs de fonctions, des pointeurs de Listes et Dictionnaires.

```
#!/usr/bin/perl
use strict;
use warnings;
use Data::Dumper;

my %hash = ();
my @array = [];
my $staticHashKey = "tty";
my $staticHashKey2 = "tty2";

if(!exists $hash{$staticHashKey}) {
    $hash{$staticHashKey} = [];
    my $tmpArray = \@{$hash{$staticHashKey}};
    push(@{$tmpArray}, "tty8");
    push(@{$tmpArray}, "tty9");
    push(@{$tmpArray}, "tty0");
    push(@{$tmpArray}, "tty7");
}

print Dumper(\%hash);
print "\n";

if(!exists $hash{$staticHashKey2}) {
    $hash{$staticHashKey2} = [];
}

print "\n\nBy Ref\n";
my $refHashTable = \%hash;

print %$refHashTable;
printf("\nKey1 : %5s / Val1[0] : %5s", ($staticHashKey,
$hash{$staticHashKey}[0]));
my $refArrayIntoHashTable = \@{$hash{$staticHashKey}};
print "\nref : ";
print ref($hash{$staticHashKey});
print "\n";
print Dumper(\@{$refArrayIntoHashTable});
```

Analysez (vos besoins), Etudiez, Scriptez L'art d'utiliser du scripting pour un projet

```
print "\n";
print @{$refArrayIntoHashTable}[2];
print "\n";

# Exemple
print "\n\n\nExemple :\n";

sub foo() {
    my @arr = ();
    push @arr, "hello", " ", " ", "world", "\n";
    my $arf = \@arr;
    return @{$arf}; # <- here
}

my @bar = foo();
map { print; } (@bar);

print "\n";

1;
```

A force d'essayer et de jouer avec les pointeurs dans tous les sens, vous comprendrez que le bug de récupération d'un pointeur d'une liste de pointeurs de Hash avec les anti-slash montre les limites de la version 5 de Perl.

Je vous ai fourni un jeu de tests montrant la limite de Perl dans le script ci-dessus, à l'endroit commenté "<- here"

Interaction entre le Scripting et une interface graphique

Grâce au module `gtk2-perl`, nous pouvons utiliser les mêmes outils graphiques qu'avec Php ou Python pour GTK+. Pour les développeurs pur Perl, il sera tout aussi simple de créer des interfaces graphiques avec Glade et de les utiliser avec Perl. Je ne présenterai donc pas d'interface dans ce langage, car je préfère laisser aux spécialistes de ce dernier ce genre de démonstrations.

F# code et script

Contenu de ce chapitre

- F# code et script..... 197
 - Pourquoi utiliser ce langage..... 198
 - Afficher simplement 'Hello World'..... 199
 - Comment utiliser les scripts par la console..... 200
 - Aller plus loin avec le Scripting en mode console..... 202
 - Interaction entre le Scripting et une interface graphique..... 206

Pourquoi utiliser ce langage

Un langage fonctionnel est utilisé pour certains calculs mathématiques dans des domaines spécifiques de la finance, car il existe une façon de prouver et garantir qu'un calcul ne peut être faux ni altéré. L'audit de ce genre de bout de code est réalisé et s'appuie sur le principe que la déclaration d'une variable ne peut pas être changée une fois celle-ci établie. Ce sont les objets non mutables. F# définit les variables comme non mutables (avec possibilité de les rendre mutables). Cela est très intéressant mais nécessite un temps d'adaptation pour changer sa compréhension d'un programme.

Ce langage peut être utilisé soit en Scripting (Interactive Mode), soit en codage avec compilation. Posé sur le Framework DotNet, c'est peut-être son avantage, car nous pouvons à la fois créer un code compilé en DotNet (et donc réaliser des programmes complexes) et créer des scripts sans compilation.

Par exemple, en programmation procédurale, nous pouvons faire

```
declare var1 <-- 2
var1 <-- var1 + 3
print var1
```

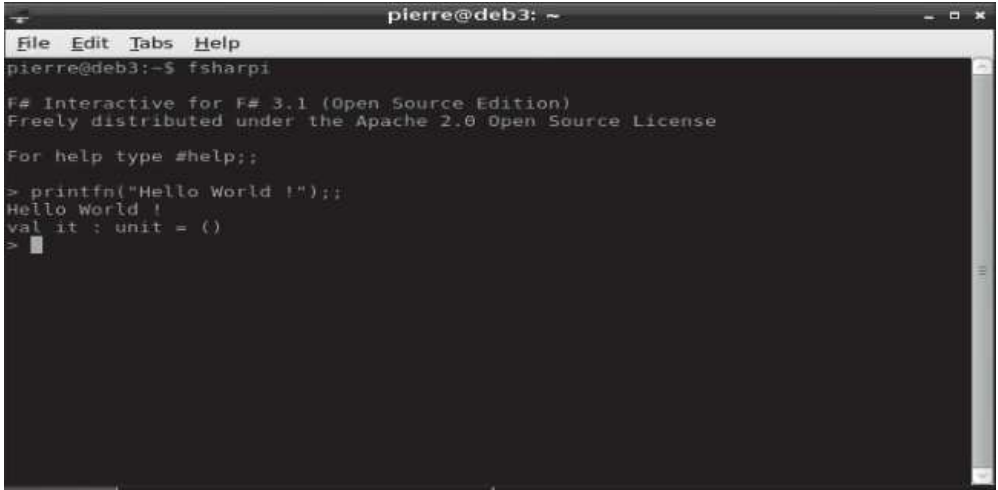
Avec F#, cette façon d'écrire doit se transformer par

```
declare var1 <-- 2
declare var2 <-- var1 + 3
print var2
```

Afficher simplement 'Hello World'

Lancer la commande interactive de Fsharp et taper

```
printfn("Hello World !");;
```



```
pierre@deb3: ~  
File Edit Tabs Help  
pierre@deb3:~$ fsharp  
F# Interactive for F# 3.1 (Open Source Edition)  
Freely distributed under the Apache 2.0 Open Source License  
For help type #help;;  
> printfn("Hello World !");;  
Hello World !  
val it : unit = ()  
> 
```

Illustration 28: Console F# Interactive sous Debian

Comment utiliser les scripts par la console

Le mode interactif de F# n'est justement qu'un mode de console.
Pour l'utiliser, en environnement UNIX, après avoir installé les packages F# (apt-get install mono-complete fsharp), il faut lancer une console, puis taper "fsharpi". Par ce mode, nous avons un prompt prêt à interpréter notre langage.
Sous Windows, cela est un peu plus compliqué. Nous devons installer F# Power Tools et utiliser une console dédiée ou alors, utiliser Visual Studio et sa console interne F# interactive.

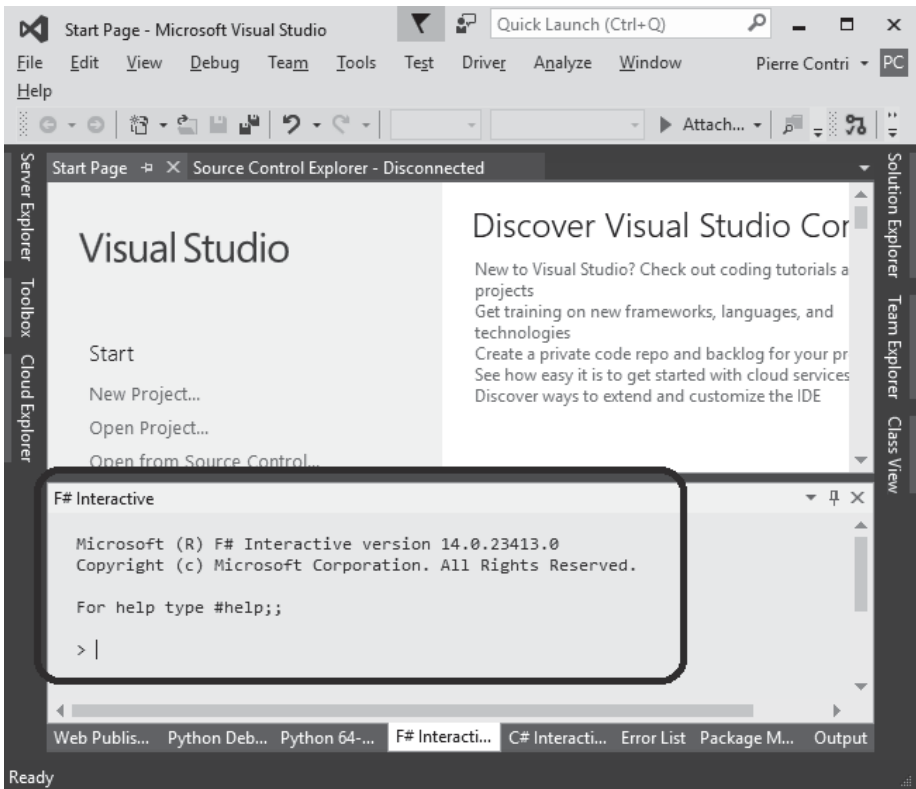


Illustration 29: F# Interactive dans Visual Studio Community

Prenons un exemple extrêmement simple pour notre premier script console.

```
open System

type Person =
    class
        val Name : string
        new(name : string) = {
            Name = name
        }
        member this.SayHello() = Console.WriteLine("Hello "
                                                    + this.Name)
    end

let p1 = new Person("Pierre")

p1.SayHello()
;;
```

Une classe contenant une variable de classe et une méthode.
On crée un objet comme en C#, puis on appelle la méthode de la même manière qu'un langage procédural.



Illustration 30: Hello World en script console

Aller plus loin avec le Scripting en mode console

Allons un peu plus loin dans ce langage et découvrons aussi la programmation fonctionnelle, pour une partie du script suivant.

Étendons le début de script précédent et observons comment hériter de la classe *Personne* en prenant appui sur les philosophies Orientées Objet, et utilisons ces classes pour définir des actions garanties par la philosophie fonctionnelle.

```
open System

type Person =
    class
        val private m_FirstName : string
        val mutable private m_NickName : string
        val private m_Age : int
        val private m_ID : Guid
        [<DefaultValue>] static val mutable private countPerson: int
        new(first : string, nickName : string, age : int) = {
            m_FirstName = first
            m_NickName = nickName
            m_Age = age
            m_ID = Guid.NewGuid()
        } // then Person.countPerson <- person.countperson + 1
        member this.id=this.m_ID
        member this.FirstName=this.m_FirstName
        member this.NickName
            with get()=this.m_NickName
            and set (value)=this.m_NickName<- value
        member this.Age=this.m_Age
        member p.sayhowdy() =
            Console.WriteLine("{0} says, 'Howdy, all' !",
                               p.FirstName)
            printfn "ok"
        override p.ToString()=String.Format("[Person: first={0},
            nickname={1}, age={2}]", p.FirstName, p.NickName, p.Age)
    end

type Employee =
    class
        inherit Person
        new() =
```

```
        {
            inherit Person("Anonymous", "Anonymous", 0)
        }
new(first : string, nickName : string, age : int) =
{
    inherit Person(first, nickName, age)
}
new(person : Person) =
{
    inherit Person(person.FirstName, person.NickName,
                    person.Age)
}
end

type TeamLeader =
class
inherit Employee
    new() =
    {
        inherit Employee("Anonymous", "Anonymous", 0)
    }
    new(first : string, nickName : string, age : int) =
    {
        inherit Employee(first, nickName, age)
    }
    new(person : Person) =
    {
        inherit Employee(person.FirstName, person.NickName,
                           person.Age)
    }
end

let people = [
    new Person("Person1", "NickName1", 33);
    new Person("Person2", "NickName2", 24);
    new Person("Person3", "NickName3", 50)
]

// ----- Total age -----
let mutable totalAge1 = 0
for p in people do
    totalAge1 <- totalAge1 + p.Age
// -----
```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
let addAges (currentTotal : int) (p : Person) = currentTotal + p.Age
let iterateAndSum (people : Person list) =
    let mutable totalAge = 0
    for p in people do
        totalAge <- addAges totalAge p
    totalAge
// -----
let addAges2 (currentTotal : int) (p : Person) = currentTotal
                                                + p.Age
let iterateAndOperate (people : Person list) (op: int -> Person ->
int) =
    let mutable totalAge = 0
    for p in people do
        totalAge <- op totalAge p
    totalAge

let totalAge2=iterateAndOperate people addAges2
// -----
let iterateAndOperate2 (seed : 'b) (coll : 'a list) (op : ('b -> 'a
-> 'b)) =
    let mutable total = seed
    for it in coll do
        total <- op total it
    total

let totalage3=iterateAndOperate2 0 people addAges2

// ----- XML transform -----
let personToXML (currentXML : string) (p : Person) =
    currentXML + "\n <person>" + p.FirstName + ""

let peopleXML = (iterateAndOperate2 " <people> " people personToXML)
+ "\n </people>"
// -----
// fun : function in line
let peopleXML2 =
    (iterateAndOperate2 " <people> " people
    (fun (curr : string) (p : Person) ->
        curr + "\n <person>" + p.FirstName + "</person>"))
    + "\n </people>"

// -----
let peopleXML3 =
    (List.fold (fun (curr) (p : Person) ->
```

Aller plus loin avec le Scripting en mode console

```

curr + "\n <person>\n  <name>" + p.FirstName + "</name>\n
<nickname>" + p.NickName + "</nickname>\n  <age>" +
p.Age.ToString() + "</age>\n  </person>") " <people> " people) + "\n
</people>\n"

// -- main --
let mainFunctionnalProgramming () =
    printfn "Sample data :\n%A" people
    printfn "Total age for people list : %d\n" totalage3
    printfn "Transform list to xml string :\n%s\n" peopleXML3

mainFunctionnalProgramming()

;;

```

Ouf, le résultat réel correspond aux attentes.

```

pierre@deb3:~/Documents$ fsharpi DemoScript.fsx
Sample data :
[[Person: first=Person1, nickname=NickName1, age=33];
 [Person: first=Person2, nickname=NickName2, age=24];
 [Person: first=Person3, nickname=NickName3, age=50]]
Total age for people list : 107

Transform list to xml string :
<people>
  <person>
    <name>Person1</name>
    <nickname>NickName1</nickname>
    <age>33</age>
  </person>
  <person>
    <name>Person2</name>
    <nickname>NickName2</nickname>
    <age>24</age>
  </person>
  <person>
    <name>Person3</name>
    <nickname>NickName3</nickname>
    <age>50</age>
  </person>
</people>

pierre@deb3:~/Documents$

```

Illustration 31: Sortie écran console des objets en XML

Ce que l'on peut retenir de cette petite démonstration est que ce langage est vraiment très intéressant et très puissant. Si on le choisit, on ne sera pas déçu, mais il faudra fournir un peu plus de raisonnement qu'un langage multi paradigme de par son approche purement fonctionnelle. Cela vaut vraiment le coup de s'y intéresser. Pour ma part, j'ai travaillé 6 mois complets avec F#, c'était vraiment excellent.

Interaction entre le Scripting et une interface graphique

Ce chapitre est un peu particulier et nous permet de nous ouvrir sur le monde des langages plus lourds, ainsi que de très haut niveau.

Il s'avère que F# est un des langages de la plate-forme DotNet, comme le C#, le VB.Net, le J#, et bien d'autres. Ce dernier, basé sur le Framework Microsoft est capable d'utiliser du WinForm ou, mieux encore, du XAML. Comme nous l'avions vu avec PowerShell qui dispose aussi de l'accès à la plate-forme DotNet, F# peut aussi pointer sur la partie service Web, par le biais d'un Apache ou d'un IIS. Il dessert alors le monde de l'intranet, internet, et donc est ouvert aux technologies du Web. Nous pourrions alors utiliser d'autres Frameworks par dessus le DotNet, tels que le MVC4, MVC5, ou plus encore. F# a la possibilité de s'interfacer avec un Framework de base de données ou toutes autres sur couches (ex: EntityFramework, BootStrap, ...).

Nous utiliserons donc F# ici pour présenter un site web F# - MVC5 de présentation de F#. Ce dernier est opérationnel depuis 2014 dans l'Azure (migré en C# car Azure ne prenait pas en compte cette technologie), mais aussi sur un Apache2 d'une Debian7 depuis la même année pour montrer la possibilité de l'utiliser sur un monde Linux ouvert.

Fsharp Demo Présentation (<http://fsharpdemopresentation.azurewebsites.net>)

Voici le contrôleur du programme. Attention, cela pique et rend fou de voir qu'il faut si peu de lignes pour générer la partie de gestion du site.

```
namespace FSharpDemoPresentation_FsMVC5.Controllers

open System
open System.Collections.Generic
open System.Linq
open System.Web
open System.Web.Mvc
open System.Web.Mvc.Ajax

type HomeController() =
    inherit Controller()
    member this.Index () = this.View() :> ActionResult

    member this.IndexCategory (title: string) (category: string) =
        this.ViewData.Add("Title", title)
        this.ViewData.Add("ContentPartialView",
            "IndexSubCategories/" + category)
        this.View("DemoContent") :> ActionResult

    member this.GetCategory () =
        let title =
            this.Request.Params.GetValues("title").FirstOrDefault()
        let partialView = "IndexSubCategories/"
        + this.Request.Params.GetValues("category").FirstOrDefault()
        this.PartialView(partialView) :> ActionResult

    member this.Presentation () = this.View() :> ActionResult
    member this.Contact () = this.View() :> ActionResult
    member this.About () =
        this.ViewData.Add("Message",
            "This application is just a presentation "
            + "of F# programming.")
        this.View() :> ActionResult
```

La vue est somme toute assez banale et n'a pas besoin d'être présentée.

En revanche, nous pourrons faire le lien, entre le contrôleur et le Modèle qui gère les 6 chapitres. Voici le modèle, tel qu'on l'attendrait en C#.

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Web;
using System.IO;

namespace FSharpDemoPresentation.Models
{
    public class ChapterPresentation
    {
        public string View { get; set; }
        public string Title { get; set; }
        public string Description { get; set; }
        public string Picture { get; set; }
    }

    public class ChaptersPresentationModel
    {
        static private DataSet dsChapters = null;

        static internal IList<ChapterPresentation> DbChapters
        {
            get
            {
                if (dsChapters == null)
                {
                    string chaptersPath =
                        HttpContext.Current.Server.MapPath(
                            "~/App_Data/XMLChapters.xml");
                    if (File.Exists(chaptersPath))
                    {
                        dsChapters = new DataSet();
                        dsChapters.ReadXml(chaptersPath);
                    }
                }
                IList<ChapterPresentation> chaptersLists = null;

                chaptersLists = (
                    from DataRow row in
                        dsChapters.Tables["Chapter"].Rows
                    select new ChapterPresentation
                    {
                        View = row["View"] as string,
                        Title = row["Title"] as string,
```

Interaction entre le Scripting et une interface graphique

```

        Description = row["Description"] as string,
        Picture = row["Picture"] as string
    }).ToList();
    return chaptersLists;
}
}
} // end class
} // end namespace

```

Ce lien, présenté en C# n'existe pas en F#, car la philosophie fonctionnelle est implicite au niveau du modèle tel que défini.

Encore une fois, nous nous apercevons que le code est extrêmement réduit. La programmation fonctionnelle va réellement à l'essentiel.

Pour porter ce projet sur toutes les plate-formes rapidement, j'ai utilisé une base de données en XML. Cette dernière, représentée complètement dans un fichier, pourrait être améliorée ou réduite avec l'utilisation de Json ou carrément un fichier CSV chargé en dictionnaire (comme dans nos exemples Python).

```

<?xml version="1.0" encoding="utf-8" ?>
<Chapters>
  <Chapter>
    <View>IndexSubCategories/_Introduction</View>
    <Title>Introduction</Title>
    <Description>This is an introduction to F#</Description>

<Picture>~/App_Data/Pictures/FSharpDemol.Introduction.png</Picture>
  </Chapter>
  <Chapter>
    <View>IndexSubCategories/_ObjectOrientedProgramming</View>
    <Title>Object Oriented Programming</Title>
    <Description>Use F# with C# background</Description>

<Picture>~/App_Data/Pictures/FSharpDemol.ObjectOrientedProgramming.p
ng</Picture>
  </Chapter>
  <Chapter>
    <View>IndexSubCategories/_CallingExternalDll</View>
    <Title>Calling External dll (.Net)</Title>
    <Description>You can easily call and use external .Net
references</Description>

```

Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
<Picture>~/App_Data/Pictures/FSharpDemo1.CallingExternalDll.png</Picture>
</Chapter>
<Chapter>
  <View>IndexSubCategories/_FunctionalProgramming</View>
  <Title>Functional Programming</Title>
  <Description>Begin with functional programming</Description>
</Chapter>
<Picture>~/App_Data/Pictures/FSharpDemo1.FunctionalProgramming.png</Picture>
</Chapter>
<Chapter>
  <View>IndexSubCategories/_FunctionalProgrammingWithLinq</View>
  <Title>Linq Functional Programming Demo</Title>
  <Description>Begin with Functional programming</Description>
</Chapter>
<Picture>~/App_Data/Pictures/FSharpDemo1.LinqFunctionalDemo.png</Picture>
</Chapter>
<Chapter>
  <View>IndexSubCategories/_Conclusion</View>
  <Title>Conclusion</Title>
  <Description>Using F# for new specifics projects multi
platform</Description>
</Chapter>
</Chapters>
```

Avec ces trois fichiers, nous venons de réaliser un site web complet sur du MVC5 utilisant les sur-couches jQuery 1.5, Bootstrap et, bien sûr, la partie Vue du MVC.

Le langage fonctionnel démontre par différentes applications qu'il est important de l'utiliser pour accéder rapidement à de l'information et de la traiter par des chemins définis et garantis par le code non mutable.

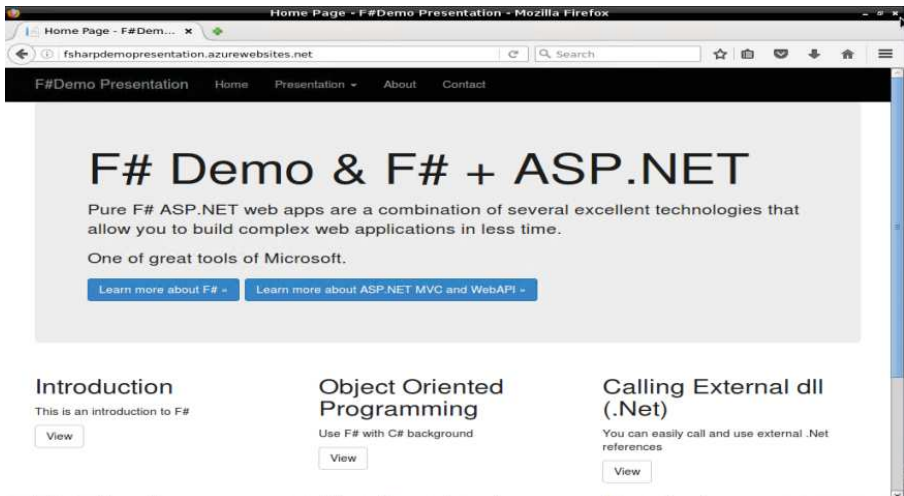


Illustration 32: Site Web en F#, MVC5

Voici maintenant le code de l'application Console utilisée aussi lors de la démonstration de F# en complément de l'application Web. Même si l'interface graphique utilise le mode console, nous pouvons naviguer dans ce programme comme un programme graphique.

Analysez (vos besoins), Etudiez, Scriptez L'art d'utiliser du scripting pour un projet

Nous allons voir que la partie Contrôleur correspond à la gestion de la console.

```
open System

[<EntryPoint>]
let main argv =

    Console.Clear()

    let mutable pageNum = 0

    while ( pageNum >= 0 ) do
        // get chapter into the dictionnary
        let chapterSelection = ChaptersData.chapterList
                                |> Seq.nth(pageNum)

        // Head Print
        // -----
        Console.Title <- "F# Demo" + " / " + chapterSelection.Key
        // Title Print
        // -----
        printfn "%s\n%s\n" chapterSelection.Key
                    ("-" .PadRight(chapterSelection.Key.Length, '-' ))

        // Execute module
        chapterSelection.Value()

        // interact with user : navigate
        pageNum <- PiTools.functionNextSection pageNum
                                ChaptersData.dictLength

    // end
    // ---
    0 // return an integer exit code
```

La gestion des chapitres se fait tel que:

```
module ChaptersData
// Chapter data
// -----

let mutable keyChapterList = seq []

/// <summary>Print the summary by using the keys from the chapters'
dictionary</summary>
/// no params, no return, just print on the standard output the
chapter list
let Summary() =

    keyChapterList |> Seq.iteri
    (
        fun countChapter chapterTitleTmp ->
            match countChapter with
            | 0 -> ()
            | _ -> printfn "%d) %s" countChapter
chapterTitleTmp
    )

/// Dictionary for all chapters
/// Key    : chapter name
/// Value  : function name of submain chapter
let chapterList = dict
[
    ("Summary", Summary); // Summary
    ("Introduction", Comments.Introduction);
    ("Object Oriented Programming",
     ObjectOrientedProgramming.mainHello);
    ("Calling external dll (.Net)",
     ExternalCalling.mainExternalCalling);
    // Calling external dll (.Net)
    ("Functionnal Programming",
     FunctionnalDemo.mainFunctionnalProgramming);
    ("Linq Functionnal Demo",
     LinqDemo.mainLinqDemo);
    ("Conclusion", Comments.Conclusion)
]

keyChapterList <- seq chapterList.Keys
let dictLength = chapterList.Count
```

Une fois ces deux gestions posées, nous utilisons les données, tel que nous l'avons fait pour l'interface web.

```
namespace FSharpDemo1.DAO

open System
open System.Configuration
open System.Data
open System.Data.Linq
open System.Collections
open System.Collections.Generic
open System.Xml
open FSharpDemo1.Model.Company
open FSharp.Configuration

type DataBase() =
    // DataBase Module
    // -----

    static let students =
        [
            new Person("Student1", "NickName1", (uint8)33);
            new Person("Student2", "NickName2", (uint8)24);
            new Person("Student3", "NickName3", (uint8)20)
        ]

    static let people =
        [
            new Person("Person1", "NickName1", (uint8)33);
            new Person("Person2", "NickName2", (uint8)24);
            new Person("Person3", "NickName3", (uint8)51)
        ]

    static let multipleSuperMario =
        [
            new Person("Boss1", "ZouperBossel", (uint8)69);
            new Person("Boss2", "ZouperBosse2", (uint8)46);
            new Person("ZouperMario3", "Plombier",
                (uint8)1664);
            new Person("ZouperMario4", "Brasseur",
                (uint8)1664)
        ]

    static let numericData =
        [|84683.0; 937.29; 37.0; 02893.0; 01320.6; 873.94349|]
```

Interaction entre le Scripting et une interface graphique

```

static let DbSet =
    let mutable dbFilePath = ""
    let ds = new DataSet()
    try
        dbFilePath <-
            AppSettings<"FSharpDemo1.DAO.dll.config">
                .DbPathFile
        if System.IO.File.Exists(dbFilePath) then
            ds.ReadXml(dbFilePath, XmlReadMode.Auto)
            |> ignore
    with
    | _ -> ()
    ds
static let dataBase = DbSet

static member NumericData =
    if dataBase.Tables.Contains("NumericData") then
        let tbl = dataBase.Tables["NumericData"].Rows
        |> Seq.cast<DataRow>
        query {
            for dbRow in tbl do
                select ( dbRow["num"].ToString() )
        }
        |> Seq.map(fun strNum ->
            float (Double.Parse(strNum,
                System.Globalization.CultureInfo.InvariantCulture)) )
        |> Seq.toArray
    else
        numericData

static member People =
    if dataBase.Tables.Contains("People") then
        let tbl = dataBase.Tables["People"].Rows
        |> Seq.cast<DataRow>
        query {
            for dbRow in tbl do
                select (
                    new Person(dbRow["first"] :?> string,
                        dbRow["nickName"] :?> string,
                        uint8 (Convert.ToUInt16(dbRow["age"].ToString())) )
                )
        }
        |> Seq.toList
    else
        people

```


Analysez (vos besoins), Etudiez, Scriptez

L'art d'utiliser du scripting pour un projet

```
static member Students =
    if dataBase.Tables.Contains("Student") then
        let tbl = dataBase.Tables["Student"].Rows
        |> Seq.cast<DataRow>

        query {
            for dbRow in tbl do
                select (
                    new Person(dbRow["first"] :?> string,
                               dbRow["nickName"] :?> string,
                               uint8 (Convert.ToInt16(dbRow["age"].ToString()))) )
        }
        |> Seq.toList
    else
        students

static member MultipleSuperMario =
    if dataBase.Tables.Contains("MultipleSuperMario") then
        let tbl = dataBase.Tables["MultipleSuperMario"].Rows
        |> Seq.cast<DataRow>

        query {
            for dbRow in tbl do
                select (
                    new Person(dbRow["first"] :?> string,
                               dbRow["nickName"] :?> string,
                               uint8 (Convert.ToInt16(dbRow["age"].ToString()))) )
        }
        |> Seq.toList
    else
        multipleSuperMario
```

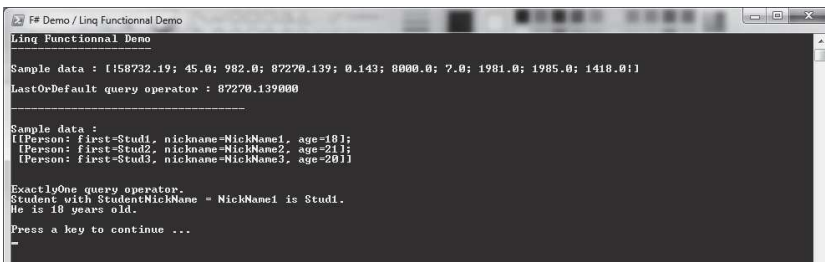


Illustration 33: Résultat d'un test d'objet en langage fonctionnel

Le résultat est assez simpliste, mais nous n'avons aucun bug de gestion d'IHM comme nous aurions pu en avoir avec un langage procédural.

A screenshot of a console window titled "F# Demo / Introduction". The window has a dark background with light-colored text. The text inside the window is as follows:

```
Introduction
-----
Why F#?
-----

--> Simple Code for Complex Problems
--> Rapid Prototyping
--> Seamless Interoperability
--> Efficient Execution
--> Reduced Complexity
--> Script or compile porgrammation (like Python)
--> Multi-Plateform
--> Open Source ; Thanks Microsoft

Press a key to continue ...
```

The window has standard Windows window controls (minimize, maximize, close) in the top right corner. A vertical scrollbar is visible on the right side of the text area.

Illustration 34: Premier chapitre du programme en mode console

Lien entre tous ces langages de script

Si vous avez un POC ou un projet complexe avec des langages multiples, ce n'est pas un problème.
L'interopérabilité technique peut être utilisée pour résoudre ces problèmes.
C'est comme une Web API dans un langage qui communique avec une autre application Web ou client lourd.

Pour tester l'interopérabilité, nous allons utiliser une bibliothèque construite en .Net avec un de nos langages de script. Nous allons démontrer que nous pouvons discuter avec un module "boîte noire" si nous avons correctement choisi notre architecture.

Contenu de ce chapitre

| | |
|--|-----|
| Lien entre tous ces langages de script..... | 219 |
| Créer une DLL et l'utiliser dans notre script..... | 220 |
| Utiliser une DLL et travailler avec en PowerShell..... | 221 |
| Utiliser une DLL et travailler avec en PHP..... | 223 |

Créer une DLL et l'utiliser dans notre script

Pour cet exercice, j'ai créé une classe nommée "PiRCommon". Cette classe accepte un paramètre à l'instanciation ou non.

Trois méthodes sont définies: SayHello (afficher "Hello World"), SayBye (afficher "Bye World"), PiR_2 (retourner le carré de la valeur du nombre PI).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace MyDll2
{
    public class PiRCommon
    {
        String personName = "World";

        public PiRCommon()
        {
        }
        public PiRCommon(String personName)
        {
            this.personName = personName;
        }
        public String SayHello()
        {
            return "Hello " + personName + " !";
        }
        public double PiR_2()
        {
            return Math.Pow(Math.PI, 2);
        }
        public String SayBye()
        {
            return "Bye " + personName + " !";
        }
    }
}
```

Compiler simplement le code et extraire la dll seule.

Utiliser une DLL et travailler avec en PowerShell

Copier la bibliothèque dans le dossier du projet PowerShell.

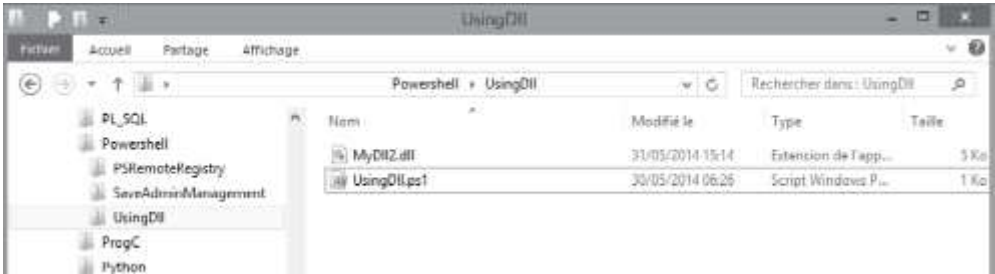


Illustration 35: Using dynamic library with PowerShell

Voici un simple code pour utiliser la dll en PowerShell v3 à v5 (j'espère plus). Pour les précédentes versions, consultez-le sur mon site web.

```
# using a dll into PowerShell Script

# assembly path
$assemblyPath = '[Insert your absolute path]\MyDll2.dll'

# ---
# Now (with PowerShell 3.0)
Add-Type -Path $assemblyPath

$obj1 = New-Object MyDll2.PiRCommon
$obj1.SayHello()
Write-Host $obj1.PiR_2()
$obj1.SayBye()

Write-Host ""

$obj2 = New-Object MyDll2.PiRCommon("Pierre")
$obj2.SayHello()
Write-Host $obj2.PiR_2()
$obj2.SayBye()
```

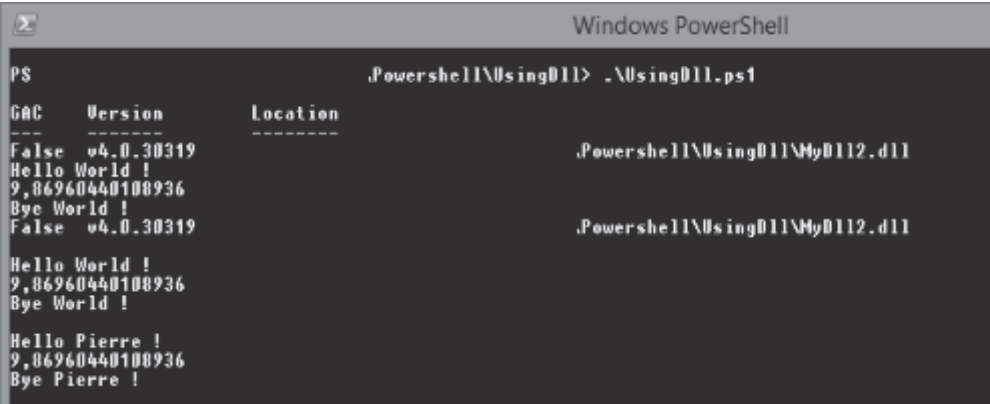


Illustration 36: Résultat de l'utilisation de la dll avec PowerShell

Utiliser une DLL et travailler avec en PHP

Dans un premier temps, en PHP, vous devez configurer le fichier "php.ini" pour accepter le chargement de librairies.

Assurez vous ensuite que le fichier PHP.ini contient ces deux lignes:

```
[COM_DOT_NET]
extension=php_com_dotnet.dll
```

Dans un second temps, vérifiez que la bibliothèque est dans le GAC (ou installez-la).

Assurez-vous que l'assembly possède un nom fort (signez le).

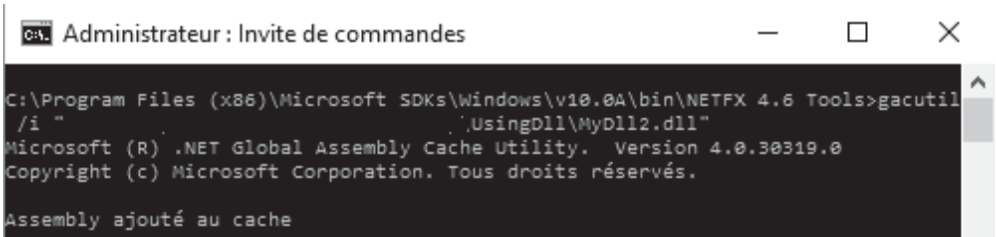


Illustration 37: Installation de la dll dans le GAC

Après cela, nous pouvons utiliser la classe COM ou DOTNET chargée dans la dll "php_com_net" fourni avec les extensions PHP.

```
<?php
// name present in the GAC
$dll_assembly_string = "MyDll2, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=4cde920f0b36e943";
$dll_class_name = "MyDll2.PirCommon";

// create the PirCommon class
$myDll2 = new DOTNET($dll_assembly_string, $dll_class_name)
    or die("error on creating ${dll_class_name}");
print($myDll2->SayHello());
print($myDll2->PiR_2());
print($myDll2->SayBye());
?>
```

Use Scripting

Contenu de ce chapitre

| | |
|--|-----|
| Use Scripting..... | 225 |
| Compiler les scripts? Pourquoi faire?..... | 226 |
| Compiler du Python en tant que ".pyc"..... | 227 |
| Compiler des scripts? OK. Utilisons d'autres langages..... | 228 |
| Convertir Python en C/C++..... | 228 |
| Utiliser DotNet C#, F# ou Java..... | 228 |
| Compatibilité..... | 229 |

Compiler les scripts? Pourquoi faire?

Si vous développez une application complète et que vous avez besoin de la distribuer, cela peut être intéressant de compiler le projet pour le déployer plus simplement avec un installateur. Avec Java, nous pourrions distribuer un seul fichier Jar qui contiendrait toutes les classes et les fichiers ressources dont nous aurions besoin pour exécuter l'application.

Nous avons aussi toute une série d'outils « installateur » qui créent des packages sous forme de « setup.exe » ou « install.msi » sous Windows. Sous Linux, la philosophie veut de plus en plus récupérer les sources d'un « git-hub » ou autre « repository ». Ou alors, appeler un script qui va réaliser l'installation du package en autonome.

La philosophie du Scripting est de fournir des programmes simples pour le déploiement ou pour une exécution dite « one shot » avec action spécifique. Ce n'est pas comme le logiciel « Libre Office » ou encore « Notepad++ » (magnifiques) qui doivent vivre dans le temps.

Avec les scripts, l'action est limitée et restreinte.

Prenons les scripts pour transformer un fichier, configurer un Pare-feu ou exécuter une action non répétable ou encore transformer le comportement d'un système.

Il n'y a donc pas d'intérêt à compiler ce genre de travail.

Le projet CarCostSimulator que j'ai proposé est un hybride, car j'ai utilisé un langage de script avec une philosophie logicielle, ce qui me permet de le faire vivre dans le temps et d'en gérer une complexité plus grande, mais cela reste marginal (laboratoires de recherche ou autres traitements qui ne sont pas liés à une contrainte de grand nombre d'utilisateur en même temps).

Compiler du Python en tant que ".pyc"

Ainsi, Python peut être compilé, et distribué comme un programme Java. Vous pouvez utiliser cela pour la production d'un logiciel. Pour se faire, une bibliothèque Python est dédiée à la compilation. Vous pouvez, grâce à ça, créer un fichier "zip" qui contient toutes les classes Python et les ressources dont le programme a besoin pour fonctionner, ou alors, créer un fichier "pyc", un fichier binaire contenant le code Python. Le cumul des deux techniques est aussi possible.

```
import py_compile;
```

Exemple avec CarCostSimulator

Voici un script PowerShell utilisé pour compiler plusieurs paquets du programme « CarCostSimulator ». Le même script peut être utilisé sur un système Linux, bien sûr ! C'est pourquoi j'ai écrit ce livre et je suis vraiment content d'avoir une interopérabilité entre systèmes.

Extrêmement puissant !

```
$absolutePathSoftware="[ENTER YOUR APPLICATION PATH]"
python $absolutePathSoftware/src/modules/zipModelizing.py

mv $absolutePathSoftware/src/modules/modelizing.zip
$absolutePathSoftware/bin/modules

python -c "import py_compile;
py_compile.compile('$absolutePathSoftware/src/modules/XmlCarManage.py', '$absolutePathSoftware/bin/modules/XmlCarManage.pyc') "

python -c "import py_compile;
py_compile.compile('$absolutePathSoftware/src/CalculVoiture.py', '$absolutePathSoftware/bin/CalculVoiture.pyc') "

cp $absolutePathSoftware/src/logging_basil.conf
$absolutePathSoftware/bin/logging_basil.conf
cp src/HMI/CalculVoiture.glade bin/HMI/CalculVoiture.glade
```

Tout comme un fichier "jar", vous pourrez exécuter la ligne de commande sur la racine du projet:

```
python bin/CalculVoiture.pyc
```

Compiler des scripts? OK. Utilisons d'autres langages

Convertir Python en C/C++

Maintenant, vous avez dit “OK, mais pas pour moi”. Alors, si ce n’est pas un problème, choisissez un autre langage que vous penserez être le meilleur pour votre projet et allez-y.

Par exemple, C++. Dans ce cas, vous pourrez être plus prêt de la machine, en évitant un « RunTime » intermédiaire. Ce qui permet aussi d’avoir un comportement plus rapide à l’exécution.

Vous pouvez aussi utiliser Jython pour Python exécuté dans la JVM, IronPython pour Python exécuté dans le Framework .Net ; dans ce cas, vous empaquetez le code python dans un bianire Java ou .Net.

En C, une commmunauté s’est créée autour de projets de transcription Python vers C et C vers Python. Cela permet de redescendre dans les couches basses des exécutable.

Cela est un exemple de travail pour le langage Python, mais il en est de même pour d’autres langages de scripts.

Utiliser DotNet C#, F# ou Java

Mais si C/C++ demande trop de temps d’adaptation, utilisez un autre langage compilé de plus haut niveau, avec Garbage Collector et des fonctions plus évoluées; par exemple, la plate-forme .Net (sur Windows et Linux).

Par contre, s’il vous plaît, pensez à la compatibilité sur les systèmes.

Il en est de même pour les bases de données. Si vous créez un nouveau projet tout objet et que vous devez intégrer par la suite une base de données, pas besoin de se restreindre à une base SQL conventionnelle. D’autres techniques de bases de données NoSQL sont disponibles, regardez avant de vous lancer.

Compatibilité

Avant de choisir votre langage de programmation en fonction du cahier des charges, prenez en compte la plate-forme ainsi que l'éco-système complet dans lequel votre application ira s'installer.

Si vous savez que le système sera juste pour des utilisateurs de Mac, développez votre solution pour Mac. N'utilisez pas .Net dans ces conditions, adaptez vous à la cible. Utilisez XCode et Objective-C ; c'est logique !

La même chose pour Windows. Si vous ne travaillez qu'avec des utilisateurs Windows, que vous avez un annuaire sur l'Active Directory, n'utilisez pas Java, ce n'est pas intéressant. Utilisez la plate-forme .Net avec la gestion des droits intégrés. DotNet devrait interagir complètement avec les systèmes Microsoft.

Si vous utilisez un système éclectique, vous devriez vous intéresser à la plate-forme Java.

Si vous travaillez avec des mathématiciens ou des scientifiques, vous pouvez être plus excentriques et aller vers Python, Scala, F#, Groovy, Smalltalk ou d'autres langages plus spécifiques.

De plus, même si tous les langages se ressemblent (d'après les dires de certains développeurs), ils ont tous une particularité, sinon nous aurions tous le même ; donc pensez à regarder les autres langages.

Scripts: aller vers la programmation fonctionnelle

Python peut aussi être utilisé pour la programmation fonctionnelle, comme le F# ou Scala.

Si vous ne voulez pas utiliser ça, utilisez la philosophie et programmez en technique fonctionnelle avec un langage multi paradigme.

Cette technique intellectuelle peut être un lien entre tout ce que vous avez vu dans ce livre.

Je pense que c'est vraiment important d'être ouvert à la programmation fonctionnelle, au vu de l'évolution de l'informatique.

Nous avons bien la programmation procédurale pendant une quarantaine d'années, puis une programmation orientée objet pendant une trentaine d'années, et je pense que l'évolution suivante sera la programmation fonctionnelle. D'où l'importance de ne pas fermer la porte sur ce savoir.

Conclusion

Utilisez les scripts pour des démonstrations, des POC, des petits projets « one shot » ou des projets complexes en accord avec la gouvernance de votre entreprise.

Je pense que vous ne pouvez pas fermer les yeux sur cette technique de programmation qui peut être utilisée pour automatiser des tâches.

Use script for Unit Test, or show evidence that a functionality is up and running.

Use script to configure a part of your operating system.

Use Scripts;

Bye World !

Scripter est excellent, mais s'il vous plaît, analysez avant de coder.

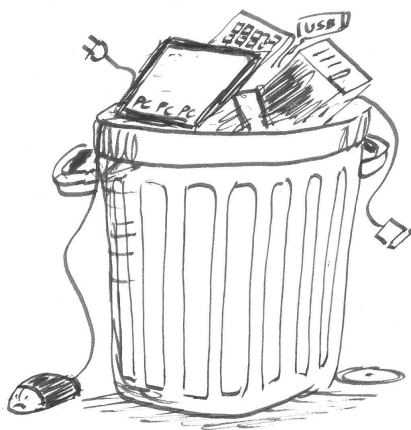
Prenez plaisir et jouez avec les Scripts!

Remerciements

Merci beaucoup à ma maman qui m'a corrigé, à plusieurs reprises, cet écrit ainsi qu'à mon papa qui a relu cet ouvrage avant impression.

Merci à mon parrain pour les dessins de couverture.

Merci à mon fils qui a joué seul, sous mon regard quand même, pendant la durée de l'écriture, ainsi qu'à ma femme qui m'a laissé du temps pour écrire.



Bibliographie

- PHP : « PHP 5 » 3e édition, L. Atkinson, CampusPress
- Python : « Python », M. Brucher, ENI
- JavaScript : « JavaScript » La référence, D. Flanagan, O'Reilly
- CSS : « CSS Avancées vers HTML5 et CSS3 », R. Goetter, Eyrolles
- PowerShell: « Scripting Avancé avec PowerShell », Kais Ayari, Eyrolles
- F#: « Programming F# 3.0 2nd Edition », Chris Smith, O'Reilly

Site Web de l'auteur

- « <http://pierre.contri.free.fr> »
- « <http://pierre.contri.free.fr/?site=programming> »



Illustration 38: Visitez le site de l'auteur

Contact

« pierre.contri@free.fr »

DevOps & IT Expert

pierre.contri@free.fr



RÉSUMÉ

Ce livre survole rapidement six langages (PHP, Python, JavaScript, PowerShell, Perl et F#) afin de montrer à quel point il est important de comprendre l'utilité du Scripting, ainsi que de donner des points d'entrées de code permettant de créer des scripts simples et rapides dans des langages divers. Tout un écrit pour promouvoir les scripts.



« <http://pierre.contri.free.fr> »

Code ISBN: 978-2-9559881-1-4



9 782955 988114