



# Developpez

*Le Mag*

Édition de février – mars 2015

Numéro 56

Magazine en ligne gratuit

Diffusion de copies conformes à l'original autorisée

Réalisation : Alexandre Pottiez & Sébastien Lataix

Rédaction : la rédaction de Developpez

Contact : magazine@redaction-developpez.com

## Sommaire

JavaScript	Page	2
CSS	Page	14
Delphi	Page	16
Java	Page	25
JavaWeb	Page	27
Libres & Open Source	Page	37
ALM	Page	48

## Éditorial

Le meilleur des magazines IT revient dans un nouveau numéro où vous retrouverez une sélection d'articles de notre site. Profitez-en bien !

La rédaction

## Article JavaScript



### JavaScript pour les Jedis, épisode I : au cœur des fonctions

Cet article parle de la boucle d'événements, du callback, des fonctions anonymes, en JavaScript

par **Wassim Chegham**

Page 2



## Article Delphi

### Comment implémenter un Singleton avec Delphi 7

Le patron de conception « Singleton » est un modèle visant à limiter l'instanciation d'une classe à un seul et unique objet.

Il est couramment utilisé pour coordonner des opérations dans un système.

par **Jérémy Laurent**

Page 16

# JavaScript



## Les derniers tutoriels et articles

# JavaScript pour les Jedis, épisode I : au cœur des fonctions

Wassim Chegham est un fervent défenseur des technologies Web front et suit de près les versions ECMAScript. Aujourd'hui, il nous propose un article sur les fonctions JavaScript et leurs aspects parfois complexes. Cet article parle de la boucle d'événements, du callback, des fonctions anonymes, etc.

## 1 Introduction

Comme vous le savez tous, 2015 est l'année de JavaScript. C'est pour cette raison que j'ai décidé de me consacrer à la rédaction d'une série d'articles concernant les fondamentaux de JavaScript, un langage très populaire, mais en même temps encore méconnu. Ces articles auront pour but de vous expliquer en détail le fonctionnement de ce langage afin de faire de vous des JavaScript Jedis.

En effet, grâce à cette série d'articles, nous allons apprendre et comprendre JavaScript afin de tirer le meilleur de ce langage. À vrai dire, j'en ai un peu marre de voir beaucoup de développeurs traiter ce langage de tous les noms, car ils n'arrivent pas à faire ce qu'ils ont l'habitude de produire avec leurs langages favoris. Mais en réalité, cela vient du fait que ces mêmes personnes ne prennent même pas la peine d'apprendre le langage, probablement à cause de sa syntaxe très ressemblante à Java. Par conséquent, ces développeurs finissent donc par développer une

sorte de haine contre JavaScript.



« JavaScript, tu l'apprends ou tu le quittes. »

Vous vous demandez sûrement pourquoi nous commençons cette série par les fonctions, et non pas par les objets. Eh bien ! Étant donné la nature fonctionnelle de JavaScript, pour moi, maîtriser cet aspect du langage fera de vous des Jedis. En effet, la plupart des développeurs via d'autres langages orientés objet essayent plus de reproduire des paradigmes propres à ces langages et font l'impasse sur les fonctions et les fermetures (closures). JavaScript est composé de ces trois concepts : les objets, les fonctions et les closures. Maîtriser JavaScript passe par l'apprentissage de cet aspect fonctionnel du langage ; et croyez-moi, le niveau de sophistication du code que vous écrirez dépend de cet apprentissage.

## 2 JavaScript est un langage fonctionnel

L'une des raisons qui font que les fonctions et la nature fonctionnelle de JavaScript sont des concepts très importants vient du fait que « la fonction » est le premier module d'exécution en JavaScript. C'est-à-dire que lorsque votre code est exécuté, il l'est au sein d'une fonction ; à l'exception des scripts qui sont interprétés lorsque le code HTML est en train d'être évalué par le navigateur. Je fais évidemment allusion à l'emblématique « `document.write()` » qui permettait de créer du DOM au runtime, obligeant le navigateur à bloquer l'interprétation du code HTML.

Revenons à nos moutons, une chose très importante à connaître sur les fonctions en JavaScript, ce sont des objets de premier ordre (first-class objects). Cela veut dire que les fonctions sont traitées comme

des objets, elles peuvent donc être :

- créées via des littérales ;
- assignées à des variables ou propriétés d'un objet ;
- passées en paramètre ;
- retournées comme résultat.

Ces fonctions peuvent aussi posséder des objets ou méthodes.

En plus d'être traitées comme des objets, les fonctions ont également une capacité spéciale : elles peuvent être invoquées (on en reparlera plus en détail dans la suite de cet article). Cette invocation est souvent effectuée de manière asynchrone. Voilà pourquoi. . .

### 3 La boucle d'événements

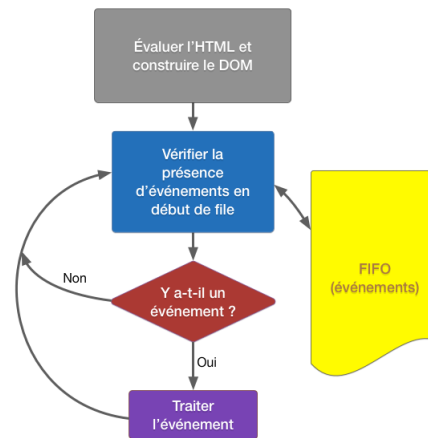
Si vous avez déjà codé une interface graphique auparavant, vous avez sûrement procédé comme ceci :

- mettre en place l'interface graphique ;
- se mettre dans une boucle d'événements en attendant qu'un événement se produise ;
- invoquer les actions à exécuter pour chaque événement.

La programmation en JavaScript — dans un navigateur — suit quasiment ces étapes, sauf que la gestion des événements est totalement prise en charge par le navigateur. Nous avons juste besoin de spécifier les actions (listeners) pour les différents événements qui peuvent être déclenchés — toujours au sein du navigateur. Ces événements sont placés dans une file au fur et à mesure qu'ils se produisent, et le navigateur traite ces événements en invoquant les actions associées. À noter que ce principe reste identique pour d'autres environnements JavaScript, par exemple NodeJS. Puisque ces événements peuvent se produire à n'importe quel moment et dans n'importe quel ordre, la gestion de ces événements, et donc l'invocation de leurs actions, se fait de manière asynchrone. Pourquoi me diriez-vous ?

Une chose à savoir, la boucle d'événements du navigateur est « monothreadée », ce qui veut dire que chaque événement se trouvant dans la file d'attente est traité dans l'ordre d'arrivée (FIFO). Les

événements sont donc traités un par un, à tour de rôle. Voici un schéma très simplifié de ce processus :



Un exemple typique du fonctionnement de cette boucle : lorsque vous bougez le curseur de votre souris dans votre page web, le navigateur détecte ces déplacements de souris et place cette série d'événements (mousemove) dans la FIFO. Ensuite, la boucle d'événements s'occupera du traitement de ces événements en exécutant les actions associées.

Ce principe d'actions — associer une fonction qui sera exécutée plus tard, lorsque l'événement se produira — illustre un mécanisme que l'on appelle les fonctions de *callback*.

### 4 Le principe de callback

Les fonctions de *callback* sont une partie très essentielle dans la bonne compréhension de JavaScript. Explorons ce point. . .

Dans la partie précédente, nous avons dit que l'on pouvait associer des actions aux différents événements se produisant au sein du navigateur. Voici un exemple d'une action :

```

1 function executerAuChargement () { /* ...
   */ };
2 window.onload = executerAuChargement;
  
```

Ici, nous avons attaché une action (listener) qui sera exécutée lorsque la page est complètement chargée par le navigateur. Parce que les fonctions sont des « first-class objects », nous pouvons affecter cette fonction à la propriété **onload** de l'objet window (représentant le navigateur).

Aussi, nous avons dit que les fonctions peuvent être passées en paramètre. Ce qui veut dire que l'on peut écrire ceci :

```

1 function dire(callback) {
2   alert(callback());
3 }
4
  
```

```

5 function bonjour () {
6   return "Bonjour, JavaScript!";
7 }
8
9 function executerAuChargement () {
10  return dire(bonjour)
11 }
12
13 window.onload = executerAuChargement;
  
```

Voilà un exemple intéressant ! Nous pouvons aussi appeler les fonctions de *callback* dans nos propres fonctions, pas besoin qu'elles soient associées à des événements. Dans cet exemple, nous avons attaché une fonction de *callback* à l'événement *onload*. Lorsque la boucle d'événement exécutera ce listener, à son tour il exécutera la fonction « dire » à laquelle nous avons passé la référence vers la fonction « bonjour ». Une fois exécutée, la fonction « dire » invoquera la fonction « bonjour », ce qui provoquera l'affichage du message « Bonjour, JavaScript ! ».

Mieux encore ? La nature fonctionnelle de JavaScript nous permet de créer des fonctions en tant qu'entité à part entière, comme n'importe quel autre type, et les passer directement en paramètres.

Voyons un exemple :

```

1  function dire(callback) {
2      alert(callback());
3  }
4
5  function executerAuChargement () {
6      return dire(function(){ return "
7          Bonjour, JavaScript!"; });
    }

```

```

8
9  window.onload = executerAuChargement;

```

Nous avons modifié l'exemple précédent en supprimant la déclaration de la fonction « *bonjour* ». Nous l'avons remplacée par la déclaration d'une fonction sans nom, directement dans les paramètres de la fonction « *dire* ». En JavaScript, nous appelons ce genre de fonctions, les fonctions anonymes.

## 5 Les fonctions anonymes

Étant donné la nature fonctionnelle du langage, il est possible de créer des fonctions n'importe où dans le code — là où une expression est attendue. Cela a pour avantage de rendre le code plus clair, plus compact et en plus, cela permet d'éviter de polluer l'espace de noms global avec des noms de fonctions inutiles.

En effet, une fonction anonyme est une fonction — comme les autres — déclarée en « *inline* » et qui n'a pas de nom pour la référencer. Ces fonctions anonymes sont généralement déclarées au moment où elles sont référencées.



« Pourquoi donner un nom à un chat qui ne réagit pas lorsqu'on l'appelle ? »

Généralement, en JavaScript, lorsque nous estimons qu'une fonction va être référencée dans plusieurs endroits du code, nous lui donnons un nom. Sinon, si elle est destinée à servir juste à un seul endroit, alors pas besoin qu'elle ait un nom. Ce qui nous amène aux déclarations et invocations des fonctions.

## 6 Déclaration de fonctions

En JavaScript, les fonctions sont déclarées à l'aide du mot-clé « *function* » qui crée une valeur de type fonction. Rappelez-vous que les fonctions sont des objets de premier ordre, elles peuvent être manipulées dans le langage comme n'importe quel autre type. De ce fait, il nous est possible d'écrire ceci (en reprenant l'exemple précédent) :

```

1  var dire = function(callback) {
2      alert(callback());
3  };

```

La valeur de la variable « *dire* » est une fonction ! Nous avons donc déclaré et créé une fonction anonyme référencée par la variable « *dire* ».

Toutes les fonctions déclarées ont une propriété « *name* » qui contient le nom de la fonction (de type String). Pour les fonctions anonymes, cette propriété est présente, mais elle est vide.

Lorsqu'une fonction a été déclarée avec un nom, ce dernier reste valide dans tout le contexte dans lequel cette fonction a été déclarée. Aussi, si cette fonction a été déclarée dans le contexte global, une nouvelle propriété portant le nom de la fonction est ajoutée dans le « *window* » ; cela est également vrai pour les variables ; ce qui nous amène donc à la suite de cet épisode concernant le contexte et la portée des fonctions.

## 7 Portée et contexte(s) d'une fonction

Lorsque nous déclarons une fonction, nous devons avoir à l'esprit, non seulement le contexte dans lequel cette fonction existe, mais également quelles sont les portées que cette fonction crée. Nous devons également faire très attention aux déclarations faites au sein de ces portées.

La gestion des portées en JavaScript est très différente de la plupart des autres langages dont la syntaxe a été influencée par le langage C. Plus précisément, ceux utilisant les accolades pour délimiter la portée des variables dans les blocs. Dans ces langages, chaque bloc crée son propre contexte. Ce n'est pas le cas de JavaScript.

« En JavaScript, les portées sont créées par les fonctions, et non pas par les blocs. »

La portée d'une variable créée au sein d'un bloc continue d'exister en dehors de ce dernier. Par exemple :

```

1  if (true) {
2      var foo = 'bar';
3  }
4  console.log(foo); //=> 'bar'

```

Ce simple exemple prouve bien que la portée de la variable « *foo* » est globale, car elle a été déclarée dans le contexte global qui est « *window* ». Elle n'est pas liée au bloc « *if* ». Regardons un exemple un peu

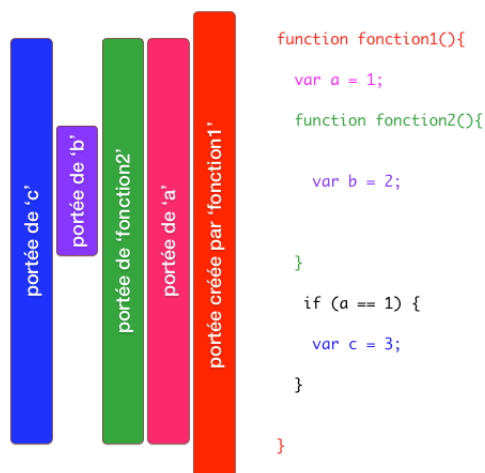
plus intéressant :

```

1  function fonction1() {
2      var a = 1;
3
4      function fonction2() {
5          var b = 2;
6      }
7
8      if (a === 1) {
9          var c = 3;
10     }
11 }
12
13 fonction1();

```

Dans cet exemple, nous avons déclaré deux fonctions, « fonction1 » et « fonction2 », et nous avons défini trois variables, a, b et c. Pour compléter cet exemple, voici un schéma plus parlant qui explique les différents contextes créés :



## 8 Invocation des fonctions

Nous avons tous invoqué des fonctions en JavaScript. Mais avez-vous déjà essayé de comprendre ce qui se passe lorsque vous invoquez une fonction ? Il existe quatre façons d'invoquer une fonction en JavaScript. Chaque type d'invocation a un effet direct sur le contexte d'exécution de cette fonction. Plus précisément, le type d'invocation agit sur le paramètre « this » passé à la fonction lors de son invocation. Écrire du code digne d'un Jedi repose sur la compréhension et la bonne maîtrise de ces mécanismes d'invocations.

Voici les différents types d'invocations en JavaScript :

1. En tant que fonction (c'est le cas le plus répandu) ;
2. En tant que méthode, ce qui permet de lier l'invocation à un objet (POO) ;
3. En tant que constructeur, ce qui permet de créer un objet (instance) ;
4. Via « apply » et « call » (voir explication plus loin).

Ce schéma résume les règles suivantes :

- la portée d'une variable existe depuis sa **déclaration**, et ce jusqu'à la fin de la fonction dans laquelle elle a été **déclarée**, peu importe l'imbrication des blocs (variables declaration hoisting) ;
- la portée d'une fonction (ne pas confondre avec une expression de fonction) est globale à toute la fonction dans laquelle elle a été déclarée (functions declaration hoisting).

Cette deuxième règle nous montre que, contrairement aux variables, les fonctions peuvent être référencées avant leur déclaration ! Oui, exactement, vous pouvez invoquer « fonction1 » avant de la déclarer. **Par contre, ce n'est pas une bonne pratique. Ce n'est pas parce que c'est autorisé par le langage qu'il faut le faire.**

Maintenant que nous avons vu comment les fonctions sont déclarées, essayons de voir comment elles peuvent être invoquées. Cela risque d'être très amusant. . .

Avant d'expliquer chaque type d'invocation, prenons quelques minutes et essayons de comprendre ce qui se passe lors d'une invocation de fonction (cela est vrai pour les cas 1, 2, et 3).

### 8.1 Arguments et paramètres

Lorsqu'une liste d'arguments est fournie lors d'une invocation de fonction, ces arguments sont affectés aux paramètres (spécifiés lors de la déclaration) de la fonction, dans le même ordre. Jusque-là rien de bien sorcier.

Maintenant, si le nombre d'arguments est différent de celui des paramètres, aucune erreur n'est déclenchée : JavaScript sait comment traiter ce cas :

- s'il y a plus d'arguments que de paramètres, alors l'excédent est ignoré ;
- s'il y a moins d'arguments que de paramètres, alors les paramètres qui n'ont pas d'argument seront mis à « undefined ».

De plus, au moment de l'invocation, deux paramètres additionnels sont passés implicitement : «

this » et « arguments ». Autrement dit, ces deux paramètres sont passés à la fonction qui est invoquée, même s'ils ne sont pas spécifiés en tant que paramètres dans la déclaration. Ils sont par la suite disponibles dans le corps de la fonction et peuvent être référencés comme n'importe quels autres vrais paramètres.

## 8.2 L'argument « arguments »

Cet objet est une sorte de collection de tous les arguments passés à la fonction.

```

1 function foo() {
2   console.log(arguments.length); //=> 3
3   console.log(arguments); //=> [1,2,'bar']
4   console.log(arguments[2]); //=> 'bar'
5   console.log(arguments.filter()); //=> undefined
6 }
7
8 foo(1,2,'bar');
```

Mais attention, même si cet objet possède un attribut « length », et permet d'accéder aux arguments via une notation comme pour un tableau « arguments[i] », ce n'est pas un tableau à proprement parler. Vous ne pouvez pas utiliser les méthodes spécifiques aux tableaux comme map() ou filter() par exemple ; autrement dit, il n'est pas possible d'itérer sur l'objet « arguments ». Cependant, il existe des astuces permettant de convertir ce type de collection en un tableau. Parmi ces astuces, nous pouvons citer :

```

1 // astuce #1
2 var args = [].slice.call(arguments, 0);
3
4 // astuce #2
5 [].forEach.call(arguments, function(x){
6   /* expression */
7 });
```

## 8.3 L'argument « this »

Cet argument est un peu particulier. Il représente l'objet qui est associé — de façon implicite — à l'invoque de la fonction. Il est aussi connu sous le terme de contexte qui est bien connu des développeurs habitués au développement orienté objet. Cependant, dans la majorité des cas, ces mêmes personnes pensent qu'en JavaScript, le mot-clé « this » représente l'instance de la classe dans laquelle la méthode est définie. Ce n'est pas vrai !

En JavaScript, il se trouve que le contexte représenté par le paramètre « this » est conditionné par la façon dont la fonction a été invoquée, et non par la façon dont elle a été définie. C'est pour cela que ce paramètre est appelé « le contexte d'invoque ».

## 8.4 Les types d'invoques

Maintenant que nous avons bien compris ce que c'est « this » et ce qu'il représente. Passons aux différents types d'invoques.

### 8.4.a Invocave en tant que fonction

Ce type d'invoque est sûrement le plus utilisé. En effet, si une fonction n'est pas invoquée en tant que méthode, constructeur ou via « apply » ou « call », alors nous sommes en présence d'invoque d'une fonction.

Cette invocave se produit lorsque la fonction est invoquée en utilisant les parenthèses et lorsque cette fonction n'est pas une propriété d'un objet. Voici un exemple plus parlant :

```

1 function foo() {};
2 foo(); //=> this === window
3 var bar = function() {};
4 bar(); //=> this === window
```

Lorsqu'il est invoqué de cette manière, le contexte d'invoque de ces fonctions est « window », le contexte global en JavaScript.

### 8.4.b Invocave en tant que méthode

Lorsqu'une fonction est une propriété d'un objet et que l'invoque se fait en appelant cette fonction, alors la fonction est invoquée en tant que méthode de cet objet. Pour illustrer cela, étudions cet exemple :

```

1 function foo() {
2   return this;
3 }
4 // invocave en tant que fonction
5 foo(); //=> this === window
6 var bar = {
7   'foo': foo
8 };
9 // invocave en tant que méthode de 'bar'
10 bar.foo(); //=> this === bar
```

Nous avons déclaré une fonction « foo » que nous invoquons de deux manières : en tant que fonction, puis en tant que méthode de l'objet « bar ». Nous constatons qu'en effet, le contexte d'invoque retourné, représenté par « this », est différent dans les deux cas. Quand invoquée en tant que fonction, « this » représente le contexte global « window ». Lorsque la fonction est invoquée en tant que méthode, « this » représente l'objet hôte de la méthode au moment de l'invoque. Cette particularité prend tout son sens en programmation orientée objet en JavaScript.

### 8.4.c Invocave en tant que constructeur

Un constructeur est une simple fonction, il n'a vraiment rien de spécial. La seule différence réside dans la façon dont il va être invoqué. Pour invoquer

une fonction en tant que constructeur, nous le précédon par le mot-clé « new ».

Invoquer une fonction en tant que constructeur est un aspect très intéressant de JavaScript, car lorsqu'un constructeur est invoqué, plusieurs choses se produisent :

- un nouvel objet vide est créé, le fameux paramètre « this » ;
- cet objet est passé au constructeur. Il devient donc le contexte d'invoation.

En l'absence d'une instruction « return » explicite dans le constructeur, cet objet est retourné par le constructeur. Prenons cet exemple :

```

1 function Foo() {
2   this.bar = function() {
3     return this;
4   }
5 }
6 // exemple 1
7 var foo = new Foo();
8 console.log(foo); //=> Foo
9 console.log(foo.bar() instanceof Foo);
  //=> true
10
11 // exemple 2
12 var foo1 = Foo();
13 console.log(typeof foo1); //=> undefined
14 console.log(typeof window.bar); //=>
  function
  
```

Nous avons déclaré un constructeur « Foo » (notez la première lettre en majuscule), celui-ci déclare une propriété « bar », qui est une méthode retournant le contexte « this » (pour l'exemple).

Dans un premier temps, nous invoquons ce constructeur avec le mot-clé « new » et nous avons bien un objet de type « Foo » qui a été créé et qui possède bien une méthode « bar ». Un petit test sur le contexte « this » dans « bar », nous prouve que c'est bien une instance de « Foo ».

Dans l'exemple 2, nous avons essayé de démontrer qu'invoquer un constructeur en tant que fonction, sans le mot-clé « new », ne donne pas du tout le résultat attendu. En effet, invoquer « Foo » en tant que fonction, exécute simplement le code de cette fonction, ce qui provoque la création de la propriété « bar » dans l'objet hôte — dans lequel « Foo » a été invoquée. Cet objet représenté par « this » dans « Foo » est bien évidemment « window », puisque la fonction « Foo » a été déclarée dans le contexte global de JavaScript. Invoquer un constructeur en omettant le mot-clé new est donc risqué, car peut potentiellement modifier d'autres objets que ceux désirés.

Jusque-là, nous avons vu que le type d'invoation d'une fonction agit directement sur le contexte d'invoation représenté par le paramètre implicite « this ». Pour les méthodes, c'est l'objet hôte ; pour les fonctions déclarées dans le contexte global, c'est le « window » ; et pour les constructeurs c'est l'instance du nouvel objet créé.

Maintenant, comment pouvons-nous faire si nous voulons forcer un contexte d'invoation particulier ? C'est bien grâce à « apply » et « call ».

#### 8.4.d Invocaton via apply() ou call()

JavaScript offre une méthode simple pour invoquer une fonction et lui spécifier explicitement un contexte d'invoation. Nous pouvons réaliser cela grâce à deux méthodes proposées par toutes les fonctions : « apply » et « call ».

Pourquoi voudrions-nous faire cela ? Prenons l'exemple d'une situation que nous rencontrons régulièrement en JavaScript : les actions — ou *callbacks* — des événements déjà abordés précédemment. Lorsque nous déclarons une fonction de *callback* sur un événement, celle-ci est invoquée lorsque l'événement en question est traité par le navigateur. Cette fonction de *callback* est invoquée avec le contexte de l'événement, c'est un comportement par défaut du navigateur.

```

1 function foo() {
2   console.log(this); //=> this === window
3   console.log(arguments); //=> 'a', 'b',
  'c'
4 }
5 var element = document.querySelector('#
  foo');
6 element.addEventListener('click',
  function(e){
7   console.log(this); //=> element
8   console.log(arguments); //=> e ===
  Event
9   foo('a', 'b', 'c');
10 });
  
```

Comme l'illustre le code ci-dessus, le contexte d'invoation de la fonction de *callback* — représentée par « this » — est celui de « window » ; la *callback* « foo » est une fonction déclarée dans le contexte global. Elle a été invoquée en tant que fonction.

Voyons maintenant le même code, mais cette fois-ci en utilisant la méthode « call ».

```

1 function foo() {
2   console.log(this); //=> this ===
  element
3   console.log(arguments); //=> 'a', 'b',
  'c'
4 }
5 var element = document.querySelector('#
  foo');
6 element.addEventListener('click',
  function(e){
7   console.log(this); //=> element
8   console.log(arguments); //=> e ===
  Event
9   foo.call(element, 'a', 'b', 'c');
10 });
  
```

En invoquant une fonction — ou méthode — avec « call », nous passons un premier paramètre qui représente le contexte d'invoation. Dans notre exemple, ce contexte est l'élément DOM qui nous intéresse. En plus du contexte d'invoation, nous

avons aussi passé une liste de paramètres à la fonction « foo ».

```

1 function foo() {
2   console.log(this); //=> this ===
   element
3   console.log(arguments); //=> a, b, c
4 }
5
6 var element = document.querySelector('#
   foo');
7 element.addEventListener('click',
   function(e) {
8
9   console.log(this); //=> element
10  console.log(arguments); //=> e ===
   Event
11
12  foo.apply(element, ['a', 'b', 'c']);
13 });

```

Avec « apply », c'est quasiment la même chose : nous passons également le contexte d'invocation en premier paramètre. Mais contrairement à « call », la méthode « apply » prend en second paramètre un tableau représentant la liste des arguments.

Voici un autre exemple plus intéressant :

```

1 function forevery(list, callback) {
2   for (var i = 0; i < list.length; i +=
   1) {
3     callback.call(list, i);
4   };
5 }
6 forevery(['a', 42, false, {}], function(
   index) {
7   console.log( "index = ", index );
8   console.log( "this = ", this );
9   console.log( "this[ index ] = ", this[
   index ] );
10  console.log( "this[ index + 1 ] = ",
   this[ index + 1 ] );
11  console.log( "this[ index - 1 ] = ",
   this[ index - 1 ] );
12
13  /*
14  *   index = 0
15  *   this = ["a", 42, false, Object]
16  *   this[ index ] = a
17  *   this[ index + 1 ] = 42
18  *   this[ index - 1 ] = undefined
19  *
20  *   index = 1
21  *   this = ["a", 42, false, Object]
22  *   this[ index ] = 42
23  *   this[ index + 1 ] = false
24  *   this[ index - 1 ] = a
25  *
26  *   etc.
27  */
28 });

```

Dans cet exemple, nous avons déclaré une fonction « forevery » qui permet d'itérer sur une liste d'éléments. Cette fonction prend en second paramètre une fonction de *callback* qui sera invoquée pour chaque élément parcouru, avec le contexte de la liste (nous aurions pu spécifier un autre contexte), et nous donne en paramètre l'index de l'élément courant.

#### 8.4.e Quelle méthode utiliser ?

Cela dépend de votre cas d'usage, et plus précisément de comment vous devez gérer vos paramètres. Si vous avez un ensemble de variables que vous devez passer en tant que paramètres, « call » serait idéale. Mais si vous avez déjà votre liste de variables sous forme de tableau (par exemple, arguments), « apply » est plus appropriée. Prenons par exemple la méthode `Math.max(n1, n2, n3...)` de JavaScript : cette méthode calcule le maximum d'une liste de valeurs qu'elle prend en paramètres.

```

1 Math.max(0, 1, 2, 3, 4, 5, 6, 7, 8, 9);
   //=> 9

```

Supposons que vous vouliez calculer le maximum (ou le minimum) d'une suite de valeurs dont vous ne connaissez pas la longueur :

```

1 function max() {
2   return (arguments.length === 0) ? 0 :
   Math.max.apply(Math, arguments);
3 }
4 max(0, 1, 2, 3, 4, 5, 6, 7, 8, 9); //=>
   9
5 max(3, 8, 32, 2, 98); //=> 98
6 max(42, 0); //=> 42
7 max(); //=> 0

```

Dans cet exemple, je viens de déclarer une fonction nommée « max » qui calcule le maximum d'une suite de nombres passés en paramètres. S'il n'y a aucun paramètre, le résultat est zéro. Je me repose donc sur le paramètre implicite « arguments » pour réaliser cette vérification. Vous comprenez tout de suite qu'utiliser « apply » a tout son sens dans cet exemple. Veuillez noter également que j'ai passé « Math » en tant que contexte d'invocation à la méthode « `Math.max()` » : je n'étais pas obligé d'en spécifier un, mais c'est plus sûr comme cela !

#### 8.5 Un mot sur bind()

En JavaScript, il existe une autre façon de contrôler le contexte d'invocation d'une fonction. Contrairement à « call » et « apply », ce contrôle est fait au moment de la déclaration de la fonction, et non à l'invocation.

Ce contrôle est rendu possible grâce à la méthode « `bind(arg)` » permettant de transformer le contexte d'invocation d'une fonction. Jetons un oeil à l'exemple suivant :

```

1 var Foo = function() {
2   this.counter = 0;
3   this.inc = function() {
4     this.counter += 1;
5     console.log(this);
6   };
7 };
8
9 var foo = new Foo();
10 var element = document.querySelector('#
   foo');
11
12 // cas #1

```



```

13 element.addEventListener('click', foo.
    inc); //=> this === element
14
15 // cas #2
16 element.addEventListener('click',
    function(){ foo.inc(); }); //=> this
    === element
17
18 // cas #3
19 element.addEventListener('click', foo.
    inc.bind(foo)); //=> this === foo
  
```

Nous avons déclaré un constructeur « Foo ». Ce dernier possède une méthode « inc », qui permet d'incrémenter un compteur « counter ». Ensuite, nous attachons un événement sur un élément DOM (disons, un bouton par exemple). À chaque clic, nous souhaitons incrémenter le compteur de « Foo ».

## 9 Résumé

Dans ce premier épisode, nous avons étudié en détail un des aspects fascinants de JavaScript : JavaScript en tant que langage fonctionnel. J'espère qu'en comprenant comment les fonctions sont traitées en interne par JavaScript, cela vous permettra de changer votre façon d'écrire du code JavaScript, en passant d'un « banal » code qui fonctionne à un code qui fonctionne, certes, mais digne d'un vrai Jedi

Dans un premier temps (cas #1), nous attachons la méthode « foo.inc » en tant qu'action sur le clic. Mais là, problème ! En effet, l'attribut « counter » n'est pas résolu ; tout simplement puisque le contexte d'invocation de « foo.inc » n'est pas l'instance « foo », mais « element » (rappelez-vous l'exemple cité plus haut avec l'« apply »).

Pour résoudre ce problème, nous aurions pu passer par une fonction anonyme (cas #2) dans laquelle nous aurions invoqué la méthode « foo.inc() ». Mais pourquoi déclarer une fonction juste pour invoquer une autre ? Il y a plus simple, nous utilisons la méthode « bind » en précisant « foo » comme contexte d'invocation (cas #3). Mais rappelez-vous que « bind » n'invoque pas la fonction en question, elle modifie simplement son contexte d'invocation.

que vous êtes.

Voilà, nous avons atteint la fin de ce premier épisode ; rendez-vous pour le second épisode dans lequel nous allons tenter de percer le mystère des fermetures en JavaScript, connu sous le nom de « closures ».

Que la force soit avec vous . . .

Retrouvez l'article de **Wassim Chegham** en ligne : [lien 1](#)

# Tutoriel La Poste IDentité Numérique - Intégration d'une API d'authentification avec Node.js

L'Internet étant aujourd'hui accessible par tous et offrant de plus en plus de services, se pose le problème de la sécurité de l'information et plus particulièrement des données personnelles mais également de la réalité des internautes avec qui nous pouvons échanger. La gestion de l'identité numérique devient un enjeu majeur. La Poste propose une solution sécurisée pour les internautes avec son système de compte vérifié en face à face : L'IDentité Numérique de La Poste.

Dans ce tutoriel, nous allons voir ce qu'est cette API, à quoi elle va servir, pour quoi s'en servir, et comment l'intégrer dans votre application pour doper vos ventes et rassurer vos interlocuteurs.

## 1 Introduction de La Poste et du langage utilisé

### 1.1 Histoire de La Poste

Société anonyme à capitaux publics depuis le 1er mars 2010, La Poste est un modèle original de groupe structuré autour de cinq branches : Services-Courrier-Colis, La Banque Postale, Réseau La Poste, GeoPost, **Numérique**.

Le Groupe est présent dans plus de 40 pays sur 4 continents.

Chaque jour, les 17 000 points de contact de La Poste, soit le 1er réseau commercial de proximité de France, accueillent 1,7 million de clients.

La Poste distribue 25 milliards d'objets par an dans le monde (lettres, imprimés publicitaires et colis), 6 jours par semaine.

En 2013, le Groupe La Poste a réalisé un chiffre d'affaires de 22,08 milliards d'euros, dont 17 % à l'international, et emploie plus de 266 000 collaborateurs.

Le Groupe La Poste, dans son plan stratégique « La Poste 2020 : Conquérir l'avenir » s'est donné pour objectif d'accélérer le développement de ses cinq branches et de conquérir de nouveaux territoires.

La Poste met le facteur humain et la confiance au cœur de la relation avec ses clients. Grâce à la convergence de ses réseaux, présente pour tous, partout et tous les jours, elle accompagne ses clients pour leur simplifier l'avenir.

La Poste est ainsi prestataire de services numériques via ses offres en ligne (MonTimbrenligne, Lettre recommandée électronique...) disponibles sur laposte.fr, ses applications mobiles, les réseaux sociaux, mais aussi son webmail laposte.net, son coffre-fort électronique Digiposte et ses filiales BtoB : Mediapost Communication et Docapost.

### 1.2 Qu'est-ce que l'IDentité Numérique ?

L'IDentité Numérique peut être définie comme un lien technologique entre une entité réelle (personne, organisme ou entreprise) et une entité virtuelle (sa ou ses représentations numériques).

Le développement et l'évolution des moyens de communication, au travers notamment de la multiplication des blogs et des réseaux sociaux, changent le rapport de l'individu à autrui. Ainsi, l'IDentité Numérique permet l'identification de l'individu en ligne et la mise en relation de celui-ci avec cet ensemble de communautés virtuelles qu'est Internet.

Dès lors, l'IDentité Numérique peut être divisée en trois catégories :

- l'identité déclarative, qui se réfère aux données saisies par l'utilisateur comme son nom, sa date de naissance, ou autres informations personnelles directement renseignées par l'individu ;
- l'identité agissante, qui est indirectement renseignée par les activités de l'utilisateur sur la toile ;
- l'identité calculée, qui résulte d'une analyse de l'identité agissante par le système, comme le nombre de communautés virtuelles dans lesquelles l'individu évolue ou bien son nombre d'amis sur les réseaux sociaux.

Le décalage ou du moins les divergences qui peuvent subsister entre l'identité déclarative et l'identité agissante soulèvent une question majeure. Qui est vraiment l'individu auquel nous avons affaire sur la toile ?

Enfin, Internet étant accessible par tous et offrant de plus en plus de services, se pose le problème de la sécurité de l'information et plus particulièrement des données personnelles. La gestion de l'identité numérique devient alors un enjeu majeur.

Voilà le principe de l'IDentité Numérique, pour plus d'informations, voici le lien Wikipédia : [lien 2](#).

Ainsi, La Poste propose une solution sécurisée pour les internautes avec son système de compte vérifié en face à face. Pour obtenir une IDentité Numérique de La Poste, l'internaute doit créer un compte ([lien 3](#)) et faire vérifier son identité par un postier afin d'activer ce compte (un postier est un agent assermenté). En utilisant son IDentité Numérique pour s'authentifier sur un site partenaire, il choisit de partager ses données vérifiées avec ce site et atteste ainsi que sa réelle identité et que son adresse postale ont été vérifiées. L'internaute garde la main

sur ses données, La Poste les sécurise et ne trace pas ses actions.

Grâce à ce service, La Poste donne à l'internaute la possibilité d'accéder à plus de services sécurisés comme la réception des lettres recommandées en ligne en toute légalité.

### 1.3 Rappel du langage

Le langage utilisé durant ce tutoriel est Node.js. Vous pourrez réviser vos bases sur la section Développez ([lien 4](#)) prévue à cet effet : Cours et tutoriels Node.js ([lien 5](#))

## 2 Découverte de l'API et de ses intérêts

### 2.1 Description de l'API

L'IDentité Numérique de La Poste : une identité vérifiée en face à face par le facteur au domicile des internautes.

C'est un service permettant à son détenteur de prouver qui il est grâce à l'identifiant et au mot de passe qu'il a choisis lors de son inscription, sans dévoiler son identité aux autres internautes.

#### 2.1.a Comment ça marche ?

Il suffit d'utiliser le bouton de connexion « La Poste Connect » intégré par les applications partenaires.

### 2.2 Usages

L'utilité de ce service est donc de fournir un identifiant numérique qui permet d'attester de son identité sur le web !

#### 2.2.a Quels sont les usages actuels ?

Le produit étant encore en bêta-test, les usages sont encore peu développés.

Aujourd'hui, l'IDentité Numérique permet de gagner du temps puisque son détenteur peut :

- recevoir des lettres recommandées en ligne (sous réserve qu'elles aient été expédiées depuis le site [boutiqueducourrier.laposte.fr](http://boutiqueducourrier.laposte.fr)) ;
- créer et activer des contrats de procurations en trois clics pour permettre à un tiers de réceptionner des courriers ou des colis à sa place ([monespaceclient.laposte.fr](http://monespaceclient.laposte.fr)).
- afficher un badge « IDentité vérifiée par La Poste » sur son profil sur site collaboratif pour donner encore plus de confiance aux internautes qui souhaitent échanger avec lui.

Le détenteur d'un compte peut également donner ses ventes sur des sites d'échanges et de location entre particuliers ([pretachanger.fr](http://pretachanger.fr) et [buzzcar](http://buzzcar)) en rassurant ses interlocuteurs. Un badge s'affiche sur son profil attestant que son identité a été vérifiée par La Poste.

Pour plus d'explications, nous vous invitons à visionner la vidéo : [lien 6](#)

## 3 Intégration de l'API

### 3.1 Clés de développement

Pour créer une application cliente, il vous faudra les clés de développement qui vous seront données après votre inscription et qui ressembleront par exemple à ça :

```
CONSUMER_KEY : _kdvsazaaphmytp4gzoi8jdbal02pfgfh5l0w12iud
CONSUMER_SECRET : _vofna5navs92smal88obo2c2s4mah1k99i5r1xls
```

\*clés factices

### 3.2 Utilisation d'une bibliothèque cliente OAuth2

Il y a de nombreux clients OAuth2 disponibles pour la plupart des langages de programmation. Afin de ne pas réinventer la roue, nous allons utiliser une bibliothèque Node.js, "simple-oauth2", dis-

ponible sur GitHub ([lien 7](#)).

La bibliothèque "simple-oauth2" fournit un ensemble de fonctionnalités utiles à la communication avec les serveurs d'identifications pour accéder aux

services de l'API Anywhere. Il s'agit notamment de rendre transparente l'utilisation du protocole d'autorisation OAuth2, et de se concentrer sur la logique du service à fournir aux utilisateurs finaux.

En supprimant la nécessité de gérer les jetons d'accès manuellement, "simple-oauth2" simplifie énormément le processus d'authentification et d'autorisation des utilisateurs de notre application cliente.

"simple-oauth2" est développé en Node.js et peut être intégré aux framework Node.js les plus courants.

Notre projet de démonstration utilisera aussi le framework javascript Express pour faciliter la gestion des routes : [lien 8](#).

Toute la documentation pour son installation et son utilisation est disponible, en anglais : [lien 9](#).

Afin de ne pas se focaliser sur une plateforme dédiée, ce document utilise la notation `base_url` pour référencer le serveur IDN cible. Par exemple, pour cibler la plateforme de recette, `base_url` est égal à l'adresse qui vous sera envoyée par e-mail dès votre demande d'adhésion au service IDN soumise, dans notre cas : [lien 10](#).

### 3.3 Installation et initialisation

Vous pouvez installer facilement les dépendances via NPM :

Bibliothèque cliente OAuth2 : `npm install simple-oauth2`

Framework javascript : `npm install express`

### 3.4 Authentification et autorisation

On peut alors utiliser "simple-oauth2" avec pour configuration minimum, la clé et le secret de l'application partenaire (CONSUMER\_KEY et CONSUMER\_SECRET).

Pour ce qui concerne l'authentification et l'autorisation, IDN possède la fonctionnalité SSO OAuth2. Il est alors possible de déléguer l'authentification de l'utilisateur au serveur IDN, puis d'utiliser la session établie.

Concernant notre application nous allons donc commencer par créer un fichier `app.js` contenant le code nécessaire à l'identification. Sa mission est de rediriger l'utilisateur vers le formulaire d'identification La Poste, le serveur d'autorisation, puis, pour la démonstration, d'afficher les données de type : nom, prénom, téléphone...

Il s'agit donc de l'application cliente. Si La Poste autorise l'accès, le serveur d'autorisation envoie un code d'autorisation à notre application cliente. Le code est ensuite échangé par un jeton d'accès. Enfin, notre application utilise ce jeton d'accès pour accéder aux données du profil de l'utilisateur par le serveur de ressources.

```
1 var express = require('express'),
2   app = express();
```

```
3
4 var oauth2 = require('simple-oauth2')({
5   clientId: " VOTRE CONSUMER_KEY ICI ",
6   clientSecret: " VOTRE CONSUMER_SECRET
7     ICI ",
8   site: '{base_url}',
9   authorizationPath: '/oauth/v2/
10     authorize',
11   tokenPath: '/oauth/v2'
12 });
13 // Authorization uri definition
14 var authorization_uri = oauth2.authCode.
15   authorizeURL({
16     redirect_uri: 'http://localhost:3000/
17     callback',
18     scope: 'openid email phone profile',
19     state: '6(ef#!)'
20 });
21 // Initial page redirecting to La Poste
22 app.get('/auth', function (req, res) {
23   res.redirect(authorization_uri);
24 });
25 // Callback service parsing the
26 // authorization token and asking for
27 // the access token
28 app.get('/callback', function (req, res)
29 {
30   var code = req.query.code;
31   console.log('/callback');
32   oauth2.authCode.getToken({
33     code: code,
34     redirect_uri: 'http://localhost:3000
35     /callback'
36   }, saveToken);
37
38   function saveToken(error, result) {
39     if (error) { console.log('Access
40       Token Error', error.message); }
41     token = oauth2.accessToken.create(
42       result);
43     console.log("token : " + token.token
44       .access_token);
45     oauth2.api('GET', '/me', {
46       access_token: token.token.
47       access_token}, function (err,
48       data) {
49       if (err) { console.log(err);res.
50         send(err);return; }
51       console.log(data);
52       res.send(data);
53     });
54   }
55 });
56 app.get('/', function (req, res) {
57   res.send('<a href="/auth">
58     Identification</a>');
59 });
60 app.listen(3000);
61 console.log('Express server started on
62   port 3000');
```

Pour cette démonstration, le jeton d'accès est affiché à l'utilisateur. Dans une application en production, l'utilisateur ne doit pas avoir accès à ce jeton, à vous de le stocker. La Poste envoie également d'autres informations en dehors du jeton d'accès.

cès comme la durée de validité du jeton d'accès.

Pour lancer l'application : `node app.js`

Si vous avez correctement configuré les variables `redirect_uri`, `clientID`, `clientSecret` et `site`, vous devez voir le lien d'identification si vous ouvrez votre

navigateur avec le port 3000 (exemple : `http://localhost:3000`). Un simple clic vous redirigera vers le site d'identification de La Poste. Une fois identifié, vous serez redirigé vers votre application qui affichera simplement les données de votre compte.

## 4 Liens utiles

- Le site de la Poste pour trouver toutes les infos utiles sur comment intégrer votre API : Développeurs IDN La Poste ([lien 11](#));
  - Le site [Developpez.com](#) pour les tutos Node.js : [lien 12](#);
  - Retrouvez le code source de l'intégration de l'API Identité Numérique dans une application sur le compte Github de BeMyApp France : [lien 13](#); .
  - Utilisez le hashtag `#LaPostedev` sur Twitter pour échanger avec La Poste sur ce service.
- Vous pouvez contacter La Poste pour toutes vos questions techniques via Contact : [lien 14](#)

Retrouvez l'article de **Jérémy Robert** en ligne : [lien 15](#)



# CSS

## Les derniers tutoriels et articles

# CSS - Icône de téléchargement animée

Voici une nouvelle astuce utilisée en CSS pour créer une icône de téléchargement animée. Dans ce tutoriel, vous mettrez en œuvre des techniques CSS plutôt simples dans lesquelles vos visiteurs pourront profiter d'une icône de téléchargement animée.

### 1 Traduction

Ce tutoriel est la traduction la plus fidèle possible du tutoriel original de Paul Underwood, Create

a animated download icon in CSS : lien 16.

### 2 Introduction

Lorsque vous faites un appel à une action sur une page Web, vous voulez normalement que les visiteurs se dirigent vers le bas pour s'inscrire à votre newsletter ou télécharger une partie de votre contenu. C'est l'appel à ces actions que vous devriez essayer de faire ressortir afin que les gens puissent se concentrer sur les endroits de votre site Internet.

Dans ce tutoriel, nous allons créer une icône de téléchargement CSS animée :



Visitez la page de démo pour voir l'animation : lien 17

### 3 Le code HTML

D'abord, nous créons le code HTML pour l'icône de téléchargement. Nous allons avoir besoin d'un lien, et à l'intérieur de cet espace, d'une zone pour l'icône de téléchargement.

```

1 <a href="http://www.example.com/download
  -content.html" class="download-icon"
  >
2   <span></span>
3   Download
4 </a>

```

### 4 Modéliser l'icône de téléchargement

Pour commencer, nous adapterons le texte au bas de l'icône. Ce texte est utilisé pour expliquer ce que cet appel action sera.

```

1 a.download-icon
2 {
3   color: #333;
4   font-size: 1.3rem;
5   font-weight: bold;
6   text-decoration: none;
7 }

```

Ensuite, nous pouvons styliser l'icône en modifiant la balise « span » pour créer une boîte rectan-

gulaire utilisant les propriétés « width » et « height », puis nous ajoutons une bordure à cet élément. En plaçant un « border-top » à transparent, cela va masquer la bordure supérieure de la vue, créant le style visuel de la boîte de téléchargement.

```

1 .download-icon span
2 {
3   display: block;
4   cursor: pointer;
5   position: relative;
6   width: 60px;
7   height: 30px;
8   margin: auto;

```

```
9 border: 10px solid #333;
10 border-top: transparent;
11 }
```

Pour créer la flèche sur l'icône de téléchargement, nous devons utiliser les sélecteurs CSS (Note du traducteur : vous pouvez également faire une recherche sur Internet dans le document de W3C, par exemple.) (lien 18) pour créer un nouvel élément à l'aide de « :before » et « :after ». Ceux-ci devront être positionnés en absolu afin que nous puissions régler la position exacte de la flèche.

```
1 .download-icon span:before,
2 .download-icon span:after
3 {
4     content: '';
5     display: block;
6     position: absolute;
7 }
```

Sur l'événement survol de « download-icon », nous allons créer la flèche et ajouter une animation d'un effet de rebond. Utilisez la propriété « animation » pour définir l'animation de rebond, ajoutez la durée de l'animation souhaitée et définissez l'animation en boucle infinie.

Sur l'élément « before », nous allons créer la ligne de la flèche ; commencez par vous positionner sur

l'icône et créez la ligne en utilisant les propriétés « width » et « height » suivies d'une couleur de fond.

Sur l'élément « after », nous pouvons alors créer un triangle en utilisant une astuce en CSS en créant une bordure et en réglant celle de gauche et de droite à « transparent ».

```
1 a.download-icon:hover span:before,
2 a.download-icon:hover span:after
3 {
4     animation: bounce .5s infinite
5         alternate;
6     -webkit-animation: bounce .5s
7         infinite alternate;
8 }
9 a.download-icon:hover span:before {
10     left: 15px;
11     top: -9px;
12     width: 10px;
13     height: 10px;
14     background: #333;
15 }
16 a.download-icon:hover span:after
17 {
18     left: 10px;
19     border-left: 10px solid transparent;
20     border-right: 10px solid transparent;
21     ;
22     border-top: 10px solid #333;
```

## 5 L'animation de rebond

Enfin, nous créons l'animation de rebond. Pour ce faire, nous avons juste besoin de changer la position de l'élément de quelques pixels. Pour cela, nous utilisons la propriété « transform » pour modifier la position « translateY » à -10px. Cela va déplacer l'élément de 10 pixels vers le haut ; à la fin de l'animation, nous devons changer le positionnement à 0 pixel.

```
1 @keyframes bounce {
2     from {
3         transform: translateY(-10px);
4     }
5     to {
```

```
7         transform: translateY(0);
8     }
9 }
10
11 @-webkit-keyframes bounce {
12     from {
13         -webkit-transform: translateY(-10px);
14     };
15     to {
16         -webkit-transform: translateY(0);
17     }
18 }
19 }
```

Démo : lien 19

## 6 Conclusion

Nous avons vu, dans ce tutoriel, comment créer simplement une icône de téléchargement animée.

N'hésitez pas à faire d'autres icônes animées et à les partager !

## 7 Liens

Vous pouvez consulter la démonstration pour avoir un aperçu du rendu dans un navigateur : lien 20

N'hésitez pas à consulter nos autres tutoriels CSS pour améliorer le visuel de vos sites : lien 21.

Retrouvez l'article de **Paul Underwood** traduit par **Danick Côté-Martel** en ligne : lien 22



# Delphi

## Les derniers tutoriels et articles

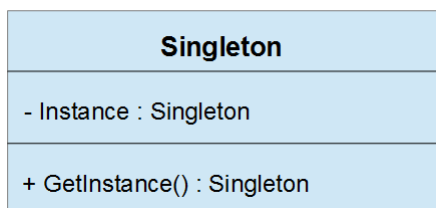
# Comment implémenter un Singleton avec Delphi 7

Le patron de conception « Singleton » est un modèle visant à limiter l'instanciation d'une classe à un seul et unique objet. Il est couramment utilisé pour coordonner des opérations dans un système.

## 1 Implémentation standard du Singleton

L'implémentation standard du Singleton consiste en une classe gérant elle-même son unique instance. Cela est réalisé au moyen d'un champ privé statique contenant l'instance et d'une méthode publique statique qui crée l'instance privée si elle n'existe pas puis la retourne.

En voici la représentation UML.



Cela ne pose aucun problème sur les dernières versions de Delphi.

Il existe d'ailleurs plusieurs discussions sur le sujet.

En voici une en particulier :

Variables globales section implémentation et objet singleton : [lien 23](#).

Et voici, le code suggéré par Paul TOTH dans cette discussion :

```

1 unit MonUnit;
2
3 interface
4

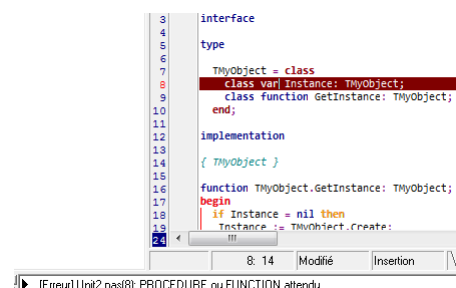
```

```

5 type
6   TMyObject = class
7     class var Instance: TMyObject;
8     class function GetInstance:
9       TMyObject;
10    end;
11 implementation
12
13 function TMyObject.GetInstance:
14   TMyObject;
15 begin
16   if Instance = nil then
17     Instance := TMyObject.Create;
18   Result := Instance;
19 end;

```

Sauf qu'avec les versions plus anciennes de Delphi, les champs statiques ne sont pas autorisés et provoquent une erreur de compilation.



Il va donc falloir stocker l'instance ailleurs.

## 2 Utilisation d'une variable globale

Puisque l'instance ne peut pas être stockée dans la classe. La seule option qui reste consiste à la placer ailleurs dans l'unité. Il faut juste savoir où.

Le choix le plus simple réside dans l'utilisation d'une variable globale accessible de n'importe où. Voici par exemple un singleton censé contenir des informations sur une configuration pour se connecter à une aide en ligne.

```

1 unit GlobalVariableSingleton;
2
3 interface
4
5 type
6
7   TOnlineHelpConfiguration = class

```



```

8 private
9   FUri : String;
10  FOffer : String;
11  FVersion : String;
12 public
13   property Uri : String read FUri
14     write FUri;
15   property Offer : String read FOffer
16     write FOffer;
17   property Version : String read
18     FVersion write FVersion;
19 end;
20
21 var
22   OnlineHelpConfiguration :
23     TOnlineHelpConfiguration;
24
25 implementation
26 uses
27   SysUtils // FreeAndNil
28   ;
29
30 initialization
31   OnlineHelpConfiguration :=

```

```

28   TOnlineHelpConfiguration.Create;
29 finalization
30   FreeAndNil(OnlineHelpConfiguration);
31
32 end.

```

L'avantage de cette méthode est qu'elle est vraiment très simple à mettre en œuvre. Elle s'accompagne malheureusement de défauts non négligeables :

- l'objet est public et peut donc potentiellement être libéré provoquant une violation d'accès lors de sa prochaine utilisation ;
- le type est accessible directement, ce qui implique que rien n'empêche la création d'une autre instance ;
- l'objet est créé même si on n'en a jamais besoin.

### 3 Utilisation d'une fonction globale

Afin de régler le problème d'accessibilité et de chargement inutile, on peut confier la création de l'objet à une fonction qui vérifiera si l'objet est instancié avant de le créer et qui fournira l'instance en retour.

```

1 unit GlobalFunctionSingleton;
2
3 interface
4
5 type
6
7   TOnlineHelpConfiguration = class
8   private
9     FUri : String;
10    FOffer : String;
11    FVersion : String;
12 public
13   property Uri : String read FUri
14     write FUri;
15   property Offer : String read FOffer
16     write FOffer;
17   property Version : String read
18     FVersion write FVersion;
19 end;
20
21 function OnlineHelpConfiguration :
22   TOnlineHelpConfiguration;
23
24 implementation
25 uses
26   SysUtils // FreeAndNil
27   ;
28
29 var
30   HiddenOnlineHelpConfiguration :
31     TOnlineHelpConfiguration;
32
33 function OnlineHelpConfiguration :
34   TOnlineHelpConfiguration;
35
36 begin
37   if not(Assigned(
38     HiddenOnlineHelpConfiguration))
39   then

```

```

31   HiddenOnlineHelpConfiguration :=
32     TOnlineHelpConfiguration.
33     Create;
34
35   Result :=
36     HiddenOnlineHelpConfiguration;
37 end;
38
39 initialization
40
41 finalization
42   FreeAndNil(
43     HiddenOnlineHelpConfiguration);
44
45 end.

```

C'est mieux, la fonction permet d'instancier l'objet uniquement lorsque c'est nécessaire bien que cela n'empêche en rien l'instanciation via le constructeur de la classe directement.

À première vue, l'objet déclaré dans la partie implémentation ne semble pas accessible, mais c'est une fausse impression, car la fonction renvoie un pointeur sur cet objet à partir duquel sa libération est toujours possible. En voyant la manière dont est créée l'instance d'objet, on est tenté de penser que sa libération ne pose pas de problème, car il sera recréé au prochain appel. Le problème c'est que l'objet pointe toujours sur une adresse en mémoire bien qu'elle ne soit plus accessible et il est toujours considéré comme assigné alors qu'il ne l'est plus. La violation d'accès est inévitable. Cette méthode pose encore problème.

Notez que l'on peut remplacer la fonction globale par une approche un peu plus orientée objet via une méthode statique. Le résultat sera rigoureusement identique. Le code ci-dessous présente uniquement les changements apportés (le reste ne bouge pas).

```

1 // Dans la partie interface
2 TOnlineHelpConfiguration = class
3 private
4   FUri : String;
5   FOffer : String;
6   FVersion : String;
7 public
8   class function GetInstance :
9     TOnlineHelpConfiguration;
10
11   property Uri : String read FUri write
12     FUri;
13   property Offer : String read FOffer
14     write FOffer;
15   property Version : String read
16     FVersion write FVersion;
17 end;

```

```

14 // Dans la partie implémentation
15
16 class function TOnlineHelpConfiguration.
17   GetInstance :
18     TOnlineHelpConfiguration;
19 begin
20   if not(Assigned(
21     HiddenOnlineHelpConfiguration))
22   then
23     HiddenOnlineHelpConfiguration :=
24       TOnlineHelpConfiguration.
25       Create;
26
27   Result :=
28     HiddenOnlineHelpConfiguration;
29 end;

```

## 4 Utilisation de l'héritage

Toute classe dans Delphi hérite obligatoirement de la classe ancêtre TObject. Cette classe ancêtre possède deux méthodes virtuelles en particulier que l'on peut surcharger pour créer ou libérer une instance. Il s'agit des méthodes NewInstance et FreeInstance. Montrer le code source de TObject ne ferait que compliquer inutilement ce tutoriel. Néanmoins, sa lecture est intéressante, car elle permet de constater que NewInstance fait appel à InitInstance pour rechercher son pointeur interne dans le gestionnaire de mémoire, créer une instance si la recherche est infructueuse ou bien retourner ce pointeur s'il a été trouvé. Elle fait donc exactement ce que nous voulons. En plus elle est statique, ce qui nous rapproche un peu plus de l'implémentation standard du singleton. FreeInstance libère la mémoire allouée, ce qui implique que si nous voulons garder toujours la même instance, il faudra appeler cette méthode uniquement lorsque cela sera réellement nécessaire.

```

1 unit ManagedByClassSingleton;
2
3 interface
4
5 type
6
7   TOnlineHelpConfiguration = class
8     private
9       FUri : String;
10      FOffer : String;
11      FVersion : String;
12     public
13       class function NewInstance : TObject
14         ; override;
15       procedure FreeInstance; override;
16
17       property Uri : String read FUri
18         write FUri;
19       property Offer : String read FOffer
20         write FOffer;
21       property Version : String read
22         FVersion write FVersion;
23
24     end;
25
26 implementation
27 uses

```

```

24   SysUtils // FreeAndNil
25   ;
26
27 var
28   HiddenOnlineHelpConfiguration :
29     TObject;
30   MustFreeInstance : Boolean;
31
32 { TOnlineHelpConfiguration }
33
34 class function TOnlineHelpConfiguration.
35   NewInstance: TObject;
36 begin
37   if not(Assigned(
38     HiddenOnlineHelpConfiguration))
39   then
40     HiddenOnlineHelpConfiguration :=
41       inherited NewInstance;
42
43   Result :=
44     HiddenOnlineHelpConfiguration;
45 end;
46
47 procedure TOnlineHelpConfiguration.
48   FreeInstance;
49 begin
50   if (MustFreeInstance) then
51     inherited FreeInstance;
52 end;
53
54 initialization
55   MustFreeInstance := False;
56
57 finalization
58   MustFreeInstance := True;
59   FreeAndNil(
60     HiddenOnlineHelpConfiguration);
61 end.

```

Ici, nous constatons que la classe peut toujours être instanciée directement, la différence réside dans le fait que la construction de l'objet va faire appel à la méthode NewInstance et donc rechercher en mémoire s'il n'y a pas déjà un pointeur sur l'objet et cela si notre propre objet caché n'est pas initialisé. Sa libération va faire de même avec la méthode FreeInstance. Sauf que la véritable libération n'est

effectuée que lorsque le booléen (inaccessible de l'extérieur, car dans la partie implémentation) passera à « Vrai », c'est-à-dire lorsque l'unité sera déchargée (lorsque le programme s'arrêtera). Nous avons

donc un objet qui est créé uniquement lorsque nous en avons besoin et qui ne sera pas libéré avant la fermeture du programme. Mission accomplie.

## 5 Singleton avancé

Pour des raisons de praticité, on peut choisir de publier un contrat à la place d'une classe. En effet, en fournissant une interface, on libère le consommateur du singleton d'une contrainte : celui-ci peut ignorer qu'il s'agit d'un singleton et l'utiliser comme n'importe quel objet (avec un Create, un try... finally et un Free). On n'a pas à se soucier de la libération d'une interface.

```

1  unit AdvancedSingleton;
2
3  interface
4
5  type
6
7      ISingleton = interface
8          ['{36771100-7CB3-4550-A444-BFC27A643DF6}']
9      end;
10
11     IOnlineHelpConfiguration = interface(
12         ISingleton)
13         ['{870E1B29-08D3-4395-BE41-B190072264E5}']
14         function GetUri : String;
15         function GetOffer : String;
16         function GetVersion : String;
17
18         procedure SetUri(Const Value :
19             String);
20         procedure SetOffer(Const Value :
21             String);
22         procedure SetVersion(Const Value :
23             String);
24
25         property Uri : String read GetUri
26             write SetUri;
27         property Offer : String read
28             GetOffer write SetOffer;
29         property Version : String read
30             GetVersion write SetVersion;
31     end;
32
33     function OnlineHelpConfiguration :
34         IOnlineHelpConfiguration;
35
36 implementation
37 uses
38     SyncObjs // TCriticalSection
39     , SysUtils // FreeAndNil
40     ;
41 type
42
43     TOnlineHelpConfiguration = class(
44         TInterfacedObject,
45         IOnlineHelpConfiguration)
46     private
47         FUri : String;
48         FOffer : String;
49         FVersion : String;
50     public

```

```

51         function GetUri : String;
52         function GetOffer : String;
53         function GetVersion : String;
54
55         procedure SetUri(Const Value :
56             String);
57         procedure SetOffer(Const Value :
58             String);
59         procedure SetVersion(Const Value :
60             String);
61
62         property Uri : String read GetUri
63             write SetUri;
64         property Offer : String read
65             GetOffer write SetOffer;
66         property Version : String read
67             GetVersion write SetVersion;
68     end;
69
70 TOnlineHelpConfigurationProvider =
71     class
72     private
73         FCriticalSection : TCriticalSection;
74         FOnlineHelpConfiguration :
75             IOnlineHelpConfiguration;
76     public
77         constructor Create;
78         destructor Destroy; override;
79
80         function GetInstance :
81             IOnlineHelpConfiguration;
82     end;
83
84 var
85     Provider :
86         TOnlineHelpConfigurationProvider;
87
88 function OnlineHelpConfiguration :
89     IOnlineHelpConfiguration;
90 begin
91     if not(Assigned(Provider)) then
92         Provider :=
93             TOnlineHelpConfigurationProvider
94                 .Create;
95
96     Result := Provider.GetInstance;
97 end;
98
99 { TOnlineHelpConfiguration }
100 function TOnlineHelpConfiguration.GetUri
101     : String;
102 begin
103     Result := FUri;
104 end;
105
106 function TOnlineHelpConfiguration.
107     GetOffer: String;
108 begin
109     Result := FOffer;
110 end;

```

```

89 function TOnlineHelpConfiguration.
    GetVersion: String;
90 begin
91     Result := FVersion;
92 end;
93
94 procedure TOnlineHelpConfiguration.
    SetUri(const Value: String);
95 begin
96     FUri := Value;
97 end;
98
99 procedure TOnlineHelpConfiguration.
    SetOffer(const Value: String);
100 begin
101     FOffer := Value;
102 end;
103
104 procedure TOnlineHelpConfiguration.
    SetVersion(const Value: String);
105 begin
106     FVersion := Value;
107 end;
108
109 { TOnlineHelpConfigurationProvider }
110
111 constructor
    TOnlineHelpConfigurationProvider.
    Create;
112 begin
113     FOnlineHelpConfiguration := nil;
114     FCriticalSection := TCriticalSection.
        Create;
115 end;
116
117 destructor
    TOnlineHelpConfigurationProvider.
    Destroy;
118 begin
119     FreeAndNil(FCriticalSection);
120     FOnlineHelpConfiguration := nil;
121     inherited;
122 end;
123
124 function
    TOnlineHelpConfigurationProvider.
    GetInstance:
    IOnlineHelpConfiguration;
125 begin
126     if not(Assigned(
        FOnlineHelpConfiguration)) then
127     begin
128         FCriticalSection.Enter;
129         try
130             // Nouvelle vérification pour gé
                rer le cas où un thread aurait

```

```

131         // attendu lors de l'instanciation
            initiale et entrerait ensuite
132         if not(Assigned(
            FOnlineHelpConfiguration))
            then
133             FOnlineHelpConfiguration :=
                TOnlineHelpConfiguration.
                    Create;
134         finally
135             FCriticalSection.Leave;
136         end;
137     end;
138
139     Result := FOnlineHelpConfiguration;
140 end;
141
142 initialization
143
144 finalization
145     FreeAndNil(Provider);
146
147 end.

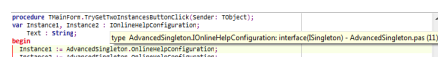
```

L'interface nous assure que le singleton ne sera jamais libéré par erreur, l'objet Provider s'occupe de créer l'instance uniquement au moment où on la sollicite et le singleton sera libéré lors du déchargement de l'unité.

Toujours dans l'optique de simplifier la consommation du singleton, la fonction globale d'accès est ici préférée à une méthode statique. Elle demeure cependant réalisable pour ceux qui voudraient se rapprocher du modèle objet.

L'objet Provider possède une section critique afin de s'assurer d'une unique instanciation, y compris dans un contexte multithread.

L'interface ISingleton n'apporte rien fonctionnellement. Elle est cependant importante pour signifier au consommateur que l'interface IOnlineHelpConfiguration répond au patron du singleton. L'image ci-dessous montre l'infobulle générée par Delphi au survol de la souris sur le type dans la déclaration des variables.



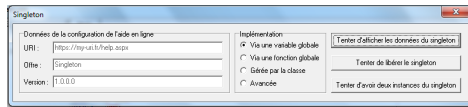
Cette implémentation du singleton est, certes, la compliquée parmi celles présentées dans ce document, mais elle est (ce n'est que mon avis) la plus robuste et la plus simple à consommer.

## 6 Consommation des différents singletons

Afin de voir fonctionnellement les différences entre les diverses implémentations du singleton, le plus simple consiste à créer une petite application de test.

L'image ci-dessous représente l'interface que j'ai choisie. Elle se présente sous la forme d'un groupe permettant la saisie des données du singleton (mais non éditable pour ne pas perturber les tests), d'un groupe de boutons radio permettant de choisir quel

type d'implémentation tester et de trois boutons pour vérifier si les implémentations répondent aux caractéristiques d'un singleton. Ce n'est qu'une présentation parmi tant d'autres et je m'accorderai volontiers avec ceux qui estimeront que ces données ne devraient pas être saisies par l'utilisateur. Cela reste cependant plus simple à comprendre sans la problématique du chargement des données.



Explications complémentaires :

Afin que les tests soient assez explicites, j'ai volontairement déclaré un booléen pour stocker le fait que des données ont déjà été stockées dans le singleton et ne pas le faire une deuxième fois (le but de la manoeuvre sera expliqué en temps voulu).

```

1 { Déclarations privées }
2 FDataExistInGlobalVariableSingleton :
   Boolean;
3 FDataExistInGlobalFunctionSingleton :
   Boolean;
4 FDataExistInManagedByClassSingleton :
   Boolean;
5 FDataExistInAdvancedSingleton : Boolean;
6
7 procedure
   GlobalVariableSingletonShowData;
8 procedure
   GlobalFunctionSingletonShowData;
9 procedure
   ManagedByClassSingletonShowData;
10 procedure AdvancedSingletonShowData;
11
12 procedure GlobalVariableSingletonFree;
13 procedure GlobalFunctionSingletonFree;
14 procedure ManagedByClassSingletonFree;
15 procedure AdvancedSingletonFree;
16
17 procedure
   GlobalVariableSingletonGetTwoInstances
18 ;
19 procedure
   GlobalFunctionSingletonGetTwoInstances
20 ;
21 procedure
   ManagedByClassSingletonGetTwoInstances
22 ;
23 procedure
   AdvancedSingletonGetTwoInstances;

```

Ces booléens sont privés, de même que les procédures à appeler pour les tests, car ces éléments ne sont pas censés être appelés ailleurs que dans la fenêtre.

Le code ci-dessous montre l'affichage des données du singleton implémenté avec une variable globale. Le booléen s'assure que les données sont affectées une seule fois.

```

1 procedure TMainForm.
   GlobalVariableSingletonShowData;
2 var Msg : String;
3 Instance : GlobalVariableSingleton.
   TOnlineHelpConfiguration;
4 begin
5 Instance := GlobalVariableSingleton.
   OnlineHelpConfiguration;
6 if not(
7 FDataExistInGlobalVariableSingleton
8 ) then
9 begin
10 Instance.Uri := Uri.Text;
11 Instance.Offer := Offer.Text;
12 Instance.Version := Version.Text;

```

```

12 FDataExistInGlobalVariableSingleton
13 := True;
14 end;
15 Msg := Format(MESSAGE_FORMAT,
16 [Instance.Uri,
17 Instance.Offer,
18 Instance.Version]);
19
20 MessageBox(Handle, PChar(Msg), PChar('
   Singleton Variable globale'), 0);
21 end;

```

Le code ci-dessous montre l'affichage des données du singleton implémenté avec une fonction globale. Le booléen s'assure que les données sont affectées une seule fois.

```

1 procedure TMainForm.
   GlobalFunctionSingletonShowData;
2 var Msg : String;
3 Instance : GlobalFunctionSingleton.
   TOnlineHelpConfiguration;
4 begin
5 Instance := GlobalFunctionSingleton.
   OnlineHelpConfiguration;
6 if not(
7 FDataExistInGlobalFunctionSingleton
8 ) then
9 begin
10 Instance.Uri := Uri.Text;
11 Instance.Offer := Offer.Text;
12 Instance.Version := Version.Text;
13
14 FDataExistInGlobalFunctionSingleton
15 := True;
16 end;
17
18 Msg := Format(MESSAGE_FORMAT,
19 [Instance.Uri,
20 Instance.Offer,
21 Instance.Version]);
22
23 MessageBox(Handle, PChar(Msg), PChar('
   Singleton Fonction globale'), 0);

```

Le code ci-dessous montre l'affichage des données du singleton implémenté à l'aide de l'héritage. Le booléen s'assure que les données sont affectées une seule fois.

```

1 procedure TMainForm.
   ManagedByClassSingletonShowData;
2 var Msg : String;
3 Instance : ManagedByClassSingleton.
   TOnlineHelpConfiguration;
4 begin
5 Instance := ManagedByClassSingleton.
   TOnlineHelpConfiguration.Create;
6 try
7 if not(
8 FDataExistInManagedByClassSingleton
9 ) then
10 begin
11 Instance.Uri := Uri.Text;
12 Instance.Offer := Offer.Text;
13 Instance.Version := Version.Text;
14
15 FDataExistInManagedByClassSingleton
16 := True;
17 end;

```

```

16     Msg := Format(MESSAGE_FORMAT,
17     [Instance.Uri,
18     Instance.Offer,
19     Instance.Version]);
20     finally
21     FreeAndNil(Instance);
22     end;
23
24     MessageBox(Handle, PChar(Msg), PChar('
Singleton géré par la classe'), 0);
25 end;

```

Le code ci-dessous montre l'affichage des données du singleton avancé. Le booléen s'assure que les données sont affectées une seule fois.

```

1 procedure TMainForm.
2     AdvancedSingletonShowData;
3 var Msg : String;
4     Instance : AdvancedSingleton.
5     IOnlineHelpConfiguration;
6 begin
7     Instance := AdvancedSingleton.
8     OnlineHelpConfiguration;
9     if not(FDataExistInAdvancedSingleton)
10    then
11    begin
12    Instance.Uri := Uri.Text;
13    Instance.Offer := Offer.Text;
14    Instance.Version := Version.Text;
15    FDataExistInAdvancedSingleton :=
16    True;
17    end;
18
19    Msg := Format(MESSAGE_FORMAT,
20    [Instance.Uri,
21    Instance.Offer,
22    Instance.Version]);
23
24    MessageBox(Handle, PChar(Msg), PChar('
Singleton avancé'), 0);
25 end;

```

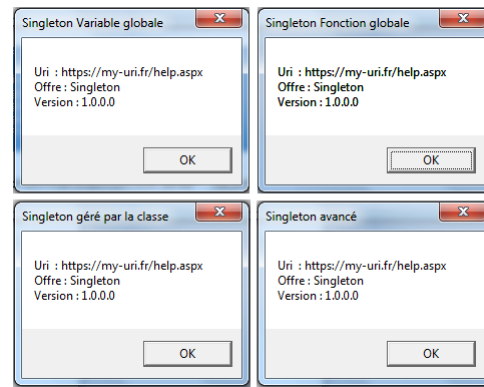
Le code ci-dessous montre le code derrière le premier bouton.

```

1 procedure TMainForm.
2     TryToShowSingletonDataButtonClick(
3     Sender: TObject);
4 begin
5     case (ImplementationChoice.ItemIndex)
6     of
7     0 : GlobalVariableSingletonShowData;
8     1 : GlobalFunctionSingletonShowData;
9     2 : ManagedByClassSingletonShowData;
10    3 : AdvancedSingletonShowData;
11    end;
12 end;

```

Ces quelques lignes permettent déjà de vérifier que les données s'affichent bien. Jusque-là, il n'y a aucun problème.



Que se passe-t-il si on essaie de libérer tous ces singletons ?

Le code ci-dessous tente de libérer le singleton implémenté avec une variable globale.

```

1 procedure TMainForm.
2     GlobalVariableSingletonFree;
3 begin
4     GlobalVariableSingleton.
5     OnlineHelpConfiguration.Free;
6 end;

```

Le code ci-dessous tente de libérer le singleton implémenté avec une fonction globale.

```

1 procedure TMainForm.
2     GlobalFunctionSingletonFree;
3 begin
4     GlobalFunctionSingleton.
5     OnlineHelpConfiguration.Free;
6 end;

```

Le code ci-dessous tente de libérer le singleton géré par la classe via l'héritage.

```

1 procedure TMainForm.
2     ManagedByClassSingletonFree;
3 var Instance : ManagedByClassSingleton.
4     TOnlineHelpConfiguration;
5 begin
6     Instance := ManagedByClassSingleton.
7     TOnlineHelpConfiguration.Create;
8     try
9     finally
10    FreeAndNil(Instance);
11    end;
12 end;

```

Le code ci-dessous tente de libérer le singleton avancé.

```

1 procedure TMainForm.
2     AdvancedSingletonFree;
3 var Instance : AdvancedSingleton.
4     IOnlineHelpConfiguration;
5 begin
6     Instance := AdvancedSingleton.
7     OnlineHelpConfiguration;
8     try
9     finally
10    { On ne libère pas une interface. On
11    peut la passer à nil si on veut
12    mais cela sera fait
13    automatiquement lors de la
14    sortie de la procédure }
15    Instance := nil;
16    end;
17 end;

```

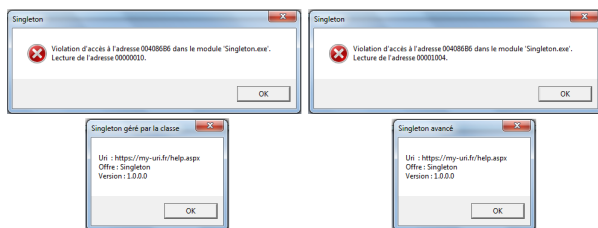
Le code ci-dessous montre le code derrière le second bouton.

```

1 procedure TMainForm.
  TryToFreeSingletonButtonClick(Sender
  : TObject);
2 begin
3   case (ImplementationChoice.ItemIndex)
4     of
5     0 : GlobalVariableSingletonFree;
6     1 : GlobalFunctionSingletonFree;
7     2 : ManagedByClassSingletonFree;
8     3 : AdvancedSingletonFree;
9   end;
end;

```

On remarque, après la libération des deux premiers singletons, que le programme devient instable. Pour les deux autres, l'affichage des données se fait bien malgré la présence des booléens empêchant le renseignement des propriétés et dont on peut, à présent, constater l'importance puisqu'elle prouve l'unicité de l'instance.



Comme vérifié précédemment, les deux dernières implémentations répondent à l'unicité du singleton. Les violations d'accès déclenchées par les deux premières suggèrent que ce n'est pas leur cas.

Le code ci-dessous tente de créer deux instances du singleton implémenté à l'aide d'une variable globale.

```

1 procedure TMainForm.
  GlobalVariableSingletonGetTwoInstances
  ;
2 var Instance1, Instance2 :
  GlobalVariableSingleton.
  TOnlineHelpConfiguration;
  Msg : String;
3 begin
4   Instance1 := GlobalVariableSingleton.
  TOnlineHelpConfiguration.Create;
5   Instance2 := GlobalVariableSingleton.
  TOnlineHelpConfiguration.Create;
6   try
7     if (Instance1 = Instance2) then Msg
8       := 'C'est la même instance.'
9     else Msg := 'Ce sont deux instances
  différentes.';
10  finally
11    FreeAndNil(Instance1);
12    FreeAndNil(Instance2);
13  end;
14  MessageBox(Handle, PChar(Msg), PChar('
  Singleton Variable globale'), 0);
15 end;

```

Le code ci-dessous tente de créer deux instances du singleton implémenté à l'aide d'une fonction globale.

```

1 procedure TMainForm.
  GlobalFunctionSingletonGetTwoInstances
  ;
2 var Instance1, Instance2 :
  GlobalFunctionSingleton.
  TOnlineHelpConfiguration;
  Msg : String;
3 begin
4   Instance1 := GlobalFunctionSingleton.
  TOnlineHelpConfiguration.Create;
5   Instance2 := GlobalFunctionSingleton.
  TOnlineHelpConfiguration.Create;
6   try
7     if (Instance1 = Instance2) then Msg
8       := 'C'est la même instance.'
9     else Msg := 'Ce sont deux instances
  différentes.';
10  finally
11    FreeAndNil(Instance1);
12    FreeAndNil(Instance2);
13  end;
14  MessageBox(Handle, PChar(Msg), PChar('
  Singleton Fonction globale'), 0);
15 end;

```

Le code ci-dessous tente de créer deux instances du singleton géré par la classe.

```

1 procedure TMainForm.
  ManagedByClassSingletonGetTwoInstances
  ;
2 var Instance1, Instance2 :
  ManagedByClassSingleton.
  TOnlineHelpConfiguration;
  Msg : String;
3 begin
4   Instance1 := ManagedByClassSingleton.
  TOnlineHelpConfiguration.Create;
5   Instance2 := ManagedByClassSingleton.
  TOnlineHelpConfiguration.Create;
6   try
7     if (Instance1 = Instance2) then Msg
8       := 'C'est la même instance.'
9     else Msg := 'Ce sont deux instances
  différentes.';
10  finally
11    FreeAndNil(Instance1);
12    FreeAndNil(Instance2);
13  end;
14  MessageBox(Handle, PChar(Msg), PChar('
  Singleton géré par la classe'), 0);
15 end;

```

Le code ci-dessous tente de créer deux instances du singleton avancé.

```

1 procedure TMainForm.
  AdvancedSingletonGetTwoInstances;
2 var Instance1, Instance2 :
  IOnlineHelpConfiguration;
  Msg : String;
3 begin
4   Instance1 := AdvancedSingleton.
  OnlineHelpConfiguration;
5   Instance2 := AdvancedSingleton.
  OnlineHelpConfiguration;
6   try
7     if (Instance1 = Instance2) then Msg :=
  'C'est la même instance.'
8     else Msg := 'Ce sont deux instances
  différentes.';
9   finally
10    FreeAndNil(Instance1);
11    FreeAndNil(Instance2);
12  end;

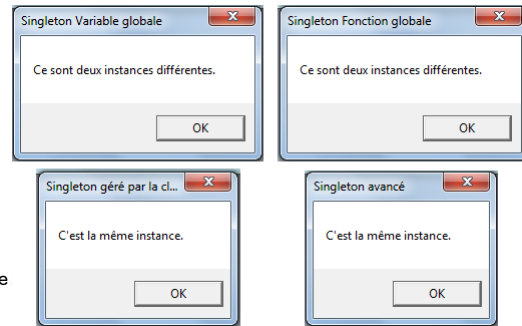
```

```
11  MessageBox(Handle, PChar(Msg), PChar('
12  end;
```

Le code ci-dessous montre le code derrière le troisième bouton.

```
1  procedure TMainForm.
2    TryGetTwoInstancesButtonClick(Sender
3    : TObject);
4  begin
5    case (ImplementationChoice.ItemIndex)
6    of
7      0 :
8        GlobalVariableSingletonGetTwoInstances
9        ;
10     1 :
11       GlobalFunctionSingletonGetTwoInstances
12       ;
13     2 :
14       ManagedByClassSingletonGetTwoInstances
15       ;
16     3 : AdvancedSingletonGetTwoInstances
17       ;
```

```
8  end;
9  end;
```



Comme prévu, l'implémentation avec une variable globale et celle avec une fonction globale ne respectent pas l'unicité alors que l'implémentation gérée par la classe et l'implémentation avancée le font.

## 7 Exemples connus dans la RTL/VCL

Lorsqu'on développe dans un IDE, un nombre important de routines et de classes sont fournies nativement. Delphi ne fait pas exception et propose quelques singletons.

C'est le cas par exemple des classes TApplication, TScreen ou encore TMouse implémentées au moyen de variables globales.

Extrait de l'unité Forms :

```
1  { Global objects }
2
3  var
4    Application: TApplication;
5    Screen: TScreen;
6
7  ...
8  implementation
```

Extrait de l'unité Controls :

```
1  var
2    Mouse: TMouse;
3
4  ...
5  implementation
6
7  ...
8  procedure InitControls;
9  var
10   UserHandle: HMODULE;
11  begin
12   ...
13   Mouse := TMouse.Create;
14   Screen := TScreen.Create(nil);
15   Application := TApplication.Create(nil);
16   ...
17  end;
18
19  ...
20  initialization
```

```
21  NewStyleControls := Lo(GetVersion) >=
22  4;
23  InitControls;
24  ...
```

On constate que les variables Application et Screen sont déclarées dans l'unité Forms et initialisées dans l'unité Controls. La variable Mouse est déclarée et initialisée dans l'unité Controls.

Une implémentation passant par une fonction globale est utilisée, entre autres, pour la classe TPrinter.

Extrait de l'unité Printers :

```
1  function Printer: TPrinter;
2
3  ...
4  implementation
5
6  uses Consts;
7
8  var
9    FPrinter: TPrinter = nil;
10
11  ...
12
13  function Printer: TPrinter;
14  begin
15    if FPrinter = nil then FPrinter :=
16      TPrinter.Create;
17    Result := FPrinter;
18  end;
19  ...
20  initialization
21
22  finalization
23    FPrinter.Free;
```

Retrouvez l'article de **Jérémy Laurent** en ligne : [lien 24](#)



# Java



## Les dernières news Java

# Java : une piste intéressante pour améliorer les types génériques - un prototype de la « generic specialization » en cours de développement

Même si Java 9 n'est pas encore prêt, une équipe de développeurs est chargée de préparer les nouvelles fonctionnalités de la version 10. Annoncé en juillet dernier, le projet Valhalla, dirigé par Brian Goetz, a pour but d'étudier et tester ces nouvelles fonctionnalités dont la publication est prévue pour 2016.

Il y a quelques jours, Goetz publia un document où il présente l'état d'avancement concernant la gestion des types génériques, l'une des caractéristiques les plus critiquées du Java puisqu'il n'est pas possible, actuellement, d'appliquer des génériques aux types primitifs.

Étant donné qu'Oracle accorde une importance primordiale à la compatibilité avec les versions précédentes, le problème soulevé de par l'introduction d'un tel système de « génériques améliorés » doit être abordé avec prudence. En effet, la difficulté est que le « système de types du langage Java n'a pas de racine unifiée », il n'y a pas de type en Java qui est à la fois un super-type d'« Object » et de « int ».

Comme l'explique Goetz, « l'un des premiers

compromis avec les génériques en Java est que ces variables ne peuvent être instanciées qu'avec les types de référence, et non pas les types primitifs [...] Nous voulons permettre aux classes génériques existantes d'être améliorées, sans avoir à les jeter et les remplacer par de nouvelles. Et en même temps ne pas forcer les programmes existants à être recompilés ou modifiés simplement pour continuer à travailler ».

Pour cela, plusieurs techniques potentielles sont en train d'être étudiées. L'une des pistes les plus prometteuses, appelée « generic specialization », consiste à continuer à représenter les types du genre `List<Integer>` et `List<String>` par `List.class` dans le runtime, tandis que les nouvelles déclarations du genre `List<int>` seront représentées par un autre type.

L'équipe du projet Valhalla est en train de préparer un prototype pour tester cette technique. Cependant, il est encore tôt pour savoir si elle permet de résoudre efficacement tous les problèmes actuels du typage générique de Java.

*Commentez la news d'**Amine Horseman** en ligne : [lien 25](#)*

# Java : De nouvelles mises à jour pour corriger 19 vulnérabilités et désactiver SSL 3.0 - il n'y aura plus de mises à jour de sécurité publiques pour la version 7

Oracle a publié de nouvelles mises à jour de sécurité pour Java afin de corriger 19 vulnérabilités et désactiver le support par défaut pour SSL 3.0, une version obsolète du protocole de communication sécurisé qui est vulnérable aux attaques.

Même si le nombre d'attaques qui exploitent les vulnérabilités de Java a été régulièrement en baisse au cours de la dernière année, les exploits Java restent parmi les principaux vecteurs d'attaque contre les utilisateurs du Web, selon le dernier rapport de sécurité de Cisco : [lien 26](#). Ces correctifs pourront donc limiter les attaques via les exploits Java.

Quatorze des 19 vulnérabilités corrigées dans Java affectent les déploiements des clients et peuvent être exploitées à partir de pages Web grâce à des applets Java malveillantes ou des applications Java Web Start. Quatre d'entre elles ont le score de sévérité maximale (10) dans le système de score de vulnérabilité commune (CVSS) et deux autres pouvant conduire à un compromis total du système sont de sévérité 9,3.

Les nouvelles mises à jour de sécurité permettent de corriger des vulnérabilités qui « *vont de la lecture et l'écriture de données locales à la prise de contrôle totale du système d'exploitation, y compris l'exécution de code arbitraire* », a déclaré John Matthew Holt, CTO de Waratek, la firme de sécurité d'appli-

cations Java, via e-mail.

Une des mises à jour de sécurité dans Java est la désactivation du protocole SSL 3.0 par défaut en réponse à la vulnérabilité de POODLE découverte en octobre : [lien 26](#). La faille permet, par des attaques de l'homme du milieu, de déchiffrer des informations sensibles comme les cookies d'authentification à partir d'une connexion chiffrée avec SSL 3.0.

POODLE a aussi affecté le protocole TLS à cause de la possibilité de forcer le passage à SSL 3.0 lorsque le client et le serveur supportent le protocole vieillissant.

Toutefois, si SSLv3 est absolument nécessaire, Oracle a indiqué dans les nouvelles de version de Java la possibilité de le réactiver.

Les versions nouvellement patchées de Java sont 5.0u81, 6u91, 7u75/7u76 et 8u31, mais les mises à jour de Java 5 et 6 ne sont disponibles que pour les clients d'Oracle avec des contrats de support à long terme.

Par ailleurs, c'est aussi la dernière mise à jour publique de sécurité pour Java 7. Les utilisateurs qui ont la mise à jour automatique activée seront migrés vers Java 8 ; et seuls les utilisateurs disposant de contrats de support à long terme seront en mesure de télécharger les prochains correctifs de sécurité de Java 7.

Commentez la news de **Michael Guilloux** en ligne : [lien 27](#)

# JavaWeb



## Les derniers tutoriels et articles

# Tutoriel sur une introduction au framework Web WebMotion

L'objectif du tutoriel est de vous faire découvrir le framework Java Web WebMotion. Le tutoriel vous permet de parcourir les notions suivantes :

- la création et le lancement d'un projet avec WebMotion ;
- les concepts (configuration, fichier de routes et les contrôleurs) ;
- la gestion des paramètres ;
- la gestion des erreurs.

Pour mettre en avant ces notions, une application de type « Hello world » est développée durant ce tutoriel.

## 1 Introduction

WebMotion est un framework Java Web : [lien 28](#). Il s'occupe des interactions entre le client (navigateur en général) et des services Java. Il repose sur JEE, en particulier sur l'API des servlets. WebMotion va vous permettre de simplifier la mise en place d'un serveur, c'est-à-dire : la conversion des paramètres, la gestion des URL... en proposant une approche KISS (*Keep it Simple, Stupid*).

D'autres frameworks Java Web existent comme Struts, Spring MVC, Play! ils comportent des faiblesses comme :

- des fichiers de configuration volumineux, par exemple pour Struts avec les fichiers de configuration en XML qui deviennent vite verbeux ;
- une architecture contraignante fixée par le framework, par exemple pour Play! il impose l'utilisation d'un serveur d'applications particulier ;
- une forte dépendance au niveau des vues, par exemple pour Struts un ensemble de tags est

nécessaire pour décrire les vues ;

- des excès d'annotations qui polluent le code Java, par exemple pour Spring MVC vous pouvez vous retrouver avec 5-6 annotations sur les méthodes de vos classes.

WebMotion s'inspire des bons concepts des différents frameworks en utilisant une syntaxe proche de Play! pour la configuration, en étant compatible sur différents serveurs d'applications (Tomcat, Jetty et Glassfish), en n'imposant pas de contraintes au niveau des vues et en utilisant à bon escient les annotations.

De plus, WebMotion s'inscrit dans une architecture orientée outils. Il n'impose pas de persistance, d'injection de dépendances... bien qu'il existe des intégrations sous forme d'extras vers Spring, Hibernate... Il est aussi compatible avec divers IDE (Eclipse, Netbeans...).

Ce tutoriel va vous permettre de lancer une première application basée sur WebMotion.

## 2 Prérequis

Le tutoriel se base en partie sur Maven et il est nécessaire d'avoir une version d'installée pour le ré-

liser complètement.

## 3 Création du projet

WebMotion utilise Maven pour vous permettre de générer un squelette de votre projet. Mais il est

possible d'utiliser un autre système de build.

```
1 $ mvn archetype:generate \
```

```

2 -DarchetypeGroupId=org.debx.
  webmotion \
3 -DarchetypeArtifactId=webmotion-
  archetype \
4 -DarchetypeVersion=2.4 \
5 -DgroupId=org.example \
6 -DartifactId=myapp \
7 -Dpackage=org.example.myapp \

```

```

8 -Dversion=1.0-SNAPSHOT \
9 -DusesExtras=N

```



Pensez à regarder la dernière version disponible.

## 4 Lancer le projet

Vous pouvez dès à présent lancer le projet sur un serveur d'applications type Tomcat, Jetty ou Glassfish. Dans le cas du tutoriel, nous allons le lancer grâce à Maven et au plugin Jetty. Pour cela, rien de plus simple, il suffit de taper :

```
1 $ mvn jetty:run
```

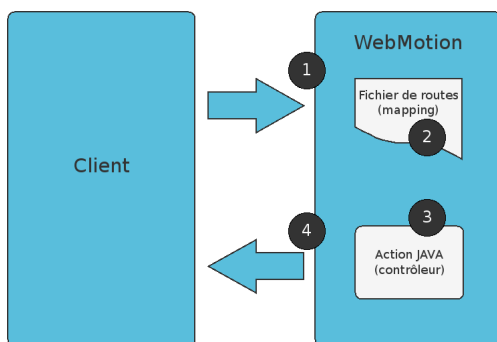
Après le démarrage du serveur, vous pouvez ouvrir l'URL suivante `http://localhost:8080/myapp/`

dans votre navigateur préféré. Vous allez voir apparaître « Hello myapp! » à l'écran. Le nom de la webapp correspond au nom de l'artefact id.

**Hello myapp!**

## 5 Principe de fonctionnement

Le schéma suivant vous présente comment fonctionne WebMotion.



1. Réception de la requête du client.
2. Recherche de l'action dans le fichier de route (fichier de mapping).
3. Exécution de l'action Java (contrôleur).
4. Retourne la réponse au client renvoyée par l'action Java (render).

Nous allons passer en revue les différents composants de WebMotion.

## 6 Configuration

Le premier composant concerne la configuration pour le serveur d'applications. WebMotion repose sur les fragments des servlets 3, ce qui lui permet de n'avoir aucune configuration dans le fichier `web.xml` :

```
1 <web-app version="3.0"
```

```

2 xmlns="http://java.sun.com/xml/ns/
  javaee"
3 xmlns:xsi="http://www.w3.org/2001/
  XMLSchema-instance"
4 xsi:schemaLocation="http://java.sun.
  com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-
  app_3_0.xsd" />

```

## 7 Le fichier de routes

Le point central du serveur avec WebMotion est le fichier de routes (nommé `mapping`). Il se trouve dans le répertoire des ressources de votre application. Ce fichier vous permet de centraliser l'ensemble des interactions sur le serveur. Remarque importante, la recherche se fait en fonction de l'ordre d'apparition des règles.

Le fichier de routes peut contenir les rubriques suivantes :

- *config* : contient la configuration (les chemins par défaut des fichiers...);
- *actions* : contient les actions liées à des URL;
- *errors* : contient les actions liées en cas d'er-

- *filters* : contient les filtres sur les actions ;
- *extensions* : inclut d'autres fichiers de routes.

Revenons sur notre fichier de routes actuel et regardons de plus près la règle suivante :

```
1 [actions]
2 * / Base.index
```

La syntaxe du fichier de routes est la suivante pour une ligne dans la rubrique d'actions :

## 8 Les contrôleurs

Les actions Java associées dans le fichier de routes sont appelées contrôleurs. La classe Java doit hériter de `WebMotionController` pour être considérée comme un contrôleur. À la fin de votre méthode, vous spécifiez la réponse pour le client, ici vers une page JSP.

```
1 public class Base extends
   WebMotionController {
2     public Render index() {
3         return renderView("index.jsp");
4     }
5 }
```

## 9 Gestion des paramètres

Nous allons modifier le fichier de routes pour gérer le cas d'un paramètre dans l'URL. Il est possible de le faire passer directement dans l'URL :

```
1 [actions]
2 * /{who} Base.index
```

ou de le faire passer en tant que paramètre :

```
1 [actions]
2 * /?who={ } Base.index
```

La conversion des types des paramètres est automatique et est gérée par `WebMotion`. Vous pouvez tout de même agir sur la conversion des paramètres en enregistrant vos propres convertisseurs. La conversion repose sur l'API `BeanUtils` d'Apache. Vous avez aussi la possibilité d'injecter des valeurs autres que celles de requêtes basées sur le type ou le nom du paramètre de la méthode.

La prochaine étape consiste à modifier l'action Java. Le paramètre est directement disponible dans la signature de la méthode ainsi :

```
1 public class Base extends
   WebMotionController {
2     public Render index(String who) {
3         return renderView("index.jsp", "
   name", who); // couple clé =
   valeur
4     }
```

1. La méthode HTTP de l'URL (GET, POST...) ici c'est « \* » pour toutes les méthodes ;
2. Le path de l'URL ;
3. L'action Java à exécuter, ici la méthode « `index` » de la classe « `Base` ».

Un autre élément de syntaxe, les commentaires dans le fichier de routes se font par un croisillon au début :

```
1 # Mon commentaire
```

Le scope par défaut des contrôleurs est de type singleton, c'est-à-dire le même contrôleur est utilisé pour toutes les requêtes. Vous pouvez modifier le cycle de vie des contrôleurs dans la section « `config` » dans le fichier de routes, pour qu'un contrôleur soit créé pour chaque requête ou qu'une même instance soit utilisée pour toutes les requêtes d'un utilisateur.

En héritant de la classe `WebMotionController`, vous disposez de l'ensemble des « `Render` » disponibles dans `WebMotion`. Un `Render` représente le retour à effectuer vers le client. Il existe plusieurs types de retours, redirections, JSON, XML, contenus, téléchargements...

```
5 }
```

Vous pouvez remarquer que le paramètre est passé à la page JSP sous la forme de clé-valeur. La page est disponible dans le répertoire « `package.views` » défini dans la section « `config` » du fichier de routes (ici `WEB-INF/org/example/myapp`). Modifiez votre page JSP pour afficher le « `name` » :

```
1 <html>
2   <head>
3     <title>myapp</title>
4   </head>
5   <body>
6     <h1>Hello ${name}!</h1>
7   </body>
8 </html>
```

Relancez votre serveur et tapez `http://localhost:8080/myapp/you` ou `http://localhost:8080/myapp/?who=you`. Vous devez maintenant voir s'afficher à l'écran « `Hello you!` ».



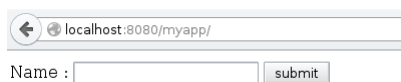
**Hello you!**

## 10 Création d'un formulaire

Finissons par ajouter un formulaire pour pouvoir saisir la valeur de notre « hello ». Pour cela, créez un fichier `form.jsp` dans le même répertoire que le fichier `index.jsp` ainsi :

```

1 <html>
2   <head>
3     <title>myapp</title>
4   </head>
5   <body>
6     <form action="/myapp/hello"
7       method="get">
8       Name : <input type="text"
9         name="who">
10      <button type="submit">submit
11    </button>
12  </form>
13 </body>
14 </html>
```



Ensuite, modifiez le fichier de mapping pour mettre le formulaire en première page et déplacer l'action Java sur l'URL `/myapp/hello`. Le mot clé « view » est un raccourci pour renvoyer l'utilisateur sur une page donnée sans passer par une action Java.

```

1 [actions]
2 * / view:
3   form.jsp
4 * /hello?who={ } Base.
5   index
```

Relancez le serveur et testez votre application. Vous pouvez saisir le nom qui vous plaît maintenant.

Pour ajouter une validation côté serveur, WebMotion s'appuie sur le standard de validation de Java (JSR 303). Vous pouvez modifier votre contrôleur pour interdire la saisie d'un nom vide :

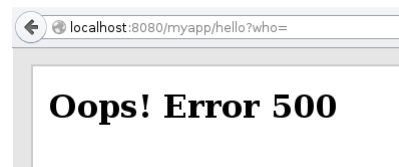
```

1 public Render index(@NotEmpty String who
2   ) {
3   return renderView("index.jsp", "name", who);
4 }
```

Si vous essayez de valider votre formulaire avec un champ vide maintenant, une page d'erreur s'affiche :

## 11 Conclusion

Nous avons vu l'approche classique de site de page en page. En ne s'occupant que de la partie serveur, WebMotion permet d'être polyvalent. Il est possible de s'intégrer dans une architecture REST en renvoyant les données sous le format JSON. Vous pouvez par exemple l'utiliser avec AngularJS pour



Pour prévenir l'utilisateur correctement en cas d'erreur, nous allons rajouter une règle dans le fichier de mapping. Il est possible, pour attraper l'erreur, avec une action associée, soit de préciser le code HTTP de l'erreur :

```

1 [errors]
2 code:500
3
4 view:error.jsp
```

soit la classe de l'exception :

```

1 [errors]
2 javax.validation.
3   ConstraintViolationException
4
5 view:error.jsp
```

Ici, une page est simplement affichée. Il serait possible de renvoyer l'ensemble des éléments au formulaire pour qu'il affiche l'erreur, mais il est conseillé d'utiliser une validation JavaScript avant d'envoyer le formulaire et de garder une validation côté serveur pour se prémunir des appels erronés avec une gestion d'erreurs globale.

```

1 <html>
2   <head>
3     <title>myapp</title>
4   </head>
5   <body>
6     <h1 style="color: red">Le champ
7       est vide.</h1>
8   </body>
9 </html>
```



**Le champ est vide.**

ne citer que celui-là.

WebMotion propose un ensemble d'extras pour développer rapidement vos applications avec :

- SiteMesh pour la décoration de page ;
- Hibernate pour l'accès à la base de données ;
- Spring pour l'injection de dépendances ;

— Shiro pour la sécurité.

Vous pouvez retrouver de nombreuses informa-

tions sur le site suivant : [lien 29](#). Je vous invite à regarder la partie démonstration qui illustre l'ensemble des fonctionnalités.

Retrouvez l'article de **Julien Ruchaud** en ligne : [lien 30](#)

# Tutorial Grails : Réaliser une application avec un framework de haute productivité

Framework Open Source dit de haute productivité dans l'écosystème Java, Grails a derrière lui une solide communauté et fait partie des projets SpringSource. Mais, que vaut-il par rapport à Groovy? Est-ce simplement un autre framework Java de plus?

Au travers de la réalisation d'une application simple, it-resto, nous pourrions entrevoir les avantages et inconvénients de cet environnement.

Cet article, en deux parties, présente dans la première la mise en place du modèle et les tests associés. Dans la seconde partie, nous verrons comment réaliser une application autour de ce modèle de données.

## 1 Présentation de Grails



Logo Grails

Projet initié en 2005 par Graeme Rocher, il avait pour but d'apporter un équivalent de Ruby on Rails au monde Java. Le nom Grails est d'ailleurs une contraction de Groovy, langage sur lequel il est basé et de Rails, l'ensemble formant Grails (Les Graals anglais, ce qui explique le logo).

Aujourd'hui, nous en sommes à la version 2.4.4 (sortie en octobre 2014).

Mais, du coup, Grails, c'est quoi exactement ?

Il s'agit d'un framework Open Source se basant sur une architecture MVC qui fonctionne sur une JVM. Sa philosophie tourne autour des points suivants :

- java-like dynamic language : les scripts Groovy sont compilés en byte-code, permettant de pro-

fiter de la puissance de Java. Il s'appuie sur son écosystème en utilisant Spring, Hibernate, etc. ;

- convention over configuration : système de configuration tendant à simplifier la vie du développeur (et limiter le code à écrire/maintenir). Par exemple, les objets du modèle auront le même nom que les tables auxquelles ils sont associés. Le mapping devient automatique ;
- DRY : éviter la redondance de code ;
- MDA : une portion du code est créée à partir du modèle, on obtient ainsi une partie du squelette de l'application. Il est donc conseillé de débiter par la génération du modèle ;
- prototypage : grâce au scaffolding, il est très facile de générer un premier prototype (même incomplet) de l'application. Il devient aisé de générer des interfaces fonctionnelles directement issues du modèle de données.

## 2 Définition du projet

Pour les besoins pédagogiques de l'article, l'application nommée it-resto, va être développée.

Mais, dans tout projet, il faut un cahier des charges. Alors, décrivons rapidement ce que nous souhaitons.

It-resto permet aux utilisateurs de participer à un événement. Pour chaque événement, il est possible de voter pour un ou plusieurs restaurants parmi

une liste précise. L'ensemble des votes permet de définir le restaurant qui accueillera l'événement.

Un cas d'utilisation simple serait, au sein d'un groupe de collègues, de permettre de sélectionner le lieu où ils choisissent de manger durant leurs pauses de midi.

L'utilisateur pourra :

- s'authentifier ;

- créer un événement ;
- participer à un événement (sans limitation de droits), en votant (une seule fois) pour un ou plusieurs restaurants dans un événement.

### 3 Premier pas avec Grails



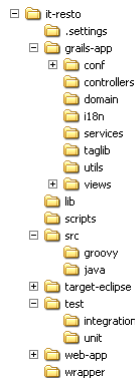
Avant d'aller plus loin, il est conseillé d'avoir installé et configuré l'environnement Grails.

Le projet est défini... nous savons où aller. Alors, allons-y!

Pour créer le projet **it-retso**, il faut lancer la commande depuis un shell (ou sous Windows, une invite de commande DOS, Babun, ConEmu, etc.) :

```
1 grails create-app it-retso
```

Un squelette d'arborescence du projet est créé dans le répertoire `it-retso`.



### 4 Mise en place du modèle

L'un des principes de base de Grails est d'être orienté modèle. La première étape est donc de générer le modèle de données (pour aller plus loin : MDA (lien 32)).

Notre modèle possède quatre éléments distincts :

- Restaurant : modélisation des restaurants à sélectionner par les utilisateurs dans un événement ;
- User : utilisateur enregistré ;
- Event : événement de base, qui englobera les votes des utilisateurs ;
- Vote : sur chaque événement, un utilisateur pourra sélectionner un ou plusieurs restaurants s'il souhaite y participer.

Afin de simplifier la compréhension de tous, un petit schéma :

L'administrateur pourra gérer (Create, Update, Read et Delete) les utilisateurs, les restaurants, les événements et les votes.

Les événements passés et leurs votes associés seront automatiquement supprimés toutes les nuits.

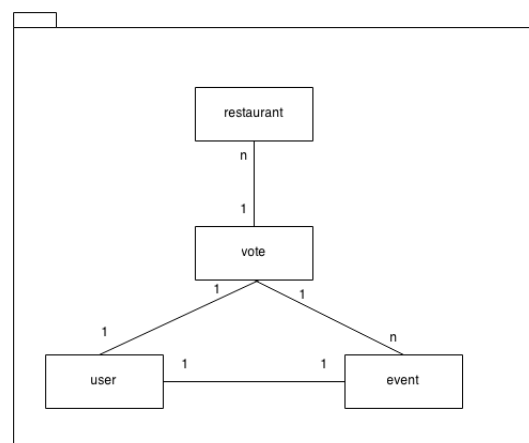
#### Arborescence de projet Grails

Le répertoire `grails-app` est le plus important. C'est ici que vont se trouver les principales ressources du projet :

- `conf` : ensemble des fichiers de configuration pour Spring, Hibernate et le projet en cours ;
- `controllers` : pas de surprise, ce sont les classes contrôleurs ;
- `views` : il s'agit des vues du projet, au format `gsp` ;
- le répertoire `lib` stockera les différentes bibliothèques du projet, qu'elles soient ajoutées via maven, ivy ou manuellement ;
- `src` : les sources java et/ou groovy ;
- pour finir, `test`... comme son nom l'indique, on y retrouve les tests unitaires, mais aussi les tests d'intégration.

Pour tester que tout s'est bien déroulé, il est possible de lancer le projet (`grails run-app`) et d'accéder au projet (lien 31).

Pour information, Grails est nativement propulsé par Tomcat.



Modèle de données

Pour aller plus vite, passons en mode interactif. Ce mode permet de travailler directement dans la console Grails, sans avoir à relancer le framework à chaque commande. De plus, les changements sont pris en compte dynamiquement.



Lancez Grails, avec la commande suivante :

```
1 grails
```

Vous êtes à présent dans la console Grails. Il faut générer les quatre éléments du modèle de données (le domain).

```
1 create-domain-class it.resto.restaurant
2 create-domain-class it.resto.user
3 create-domain-class it.resto.event
4 create-domain-class it.resto.vote
```

À chaque fois, deux fichiers sont créés :

- le premier, dans `grails-app/domain/it/resto/*`.groovy, pour la définition du domain ;
- le second, dans `test/unit/it/resto/*`Spec.groovy pour les tests unitaires ;

En fonction du besoin du projet, les différents domain doivent être complétés. Pour le domain Restaurant, on aura :

```
1 class Restaurant {
2     String name
3
4     static constraints = {
5         name blank: false, unique: true
6     }
7 }
```



Le code Groovy a une syntaxe assez proche de Java, mais un peu plus épurée. Pour en savoir un peu plus, vous pouvez consulter cette introduction à Groovy : [lien 33](#).

L'objet `it.resto.Restaurant` n'est défini que par son nom (`name`). Ensuite, dans le bloc `constraints`, on peut définir toutes les contraintes sur les différents champs.

Dans notre cas, le nom du restaurant sera non null (par défaut dans les domain Grails), non vide (`blank : false`) et unique (`unique : true`).

Bien que plus complète, la déclaration d'un User est assez similaire.

```
1 class User {
2
3     String name
4     String lastName
5     String login
6     String password
7     String email
8     boolean admin = false
9
10    String toString() {
11        return this.lastName + ' ' +
12            this.name
13    }
14
15    static constraints = {
16        name blank: false
17        lastName blank: false
18        login size: 5..15, blank: false,
19            unique: true
20        password size: 5..15, blank:
21            false
```

```
19         email email: true
20     }
21 }
```

Cette fois, les contraintes sont plus importantes. On notera surtout que la taille du login et du password doivent être comprises entre 5 et 15 caractères (`size : 5..15`). Le champ email, quant à lui, doit répondre aux contraintes d'un email (`email : true`).

Maintenant, passons à la classe `it.resto.Event`, qui devra être liée avec plusieurs votes.

```
1 class Event {
2
3     String name
4     Date eventDate
5     User owner
6     static hasMany = [votes: Vote]
7
8     String toString() {
9         return this.name;
10    }
11
12    static constraints = {
13        name blank: false
14        eventDate nullable: true
15    }
16 }
```

En regardant l'ensemble des contraintes, on se rend compte que tous les champs sont obligatoires.

La relation de type 1 ↔ N entre l'Event et les Vote est défini avec le mot clef `hasMany`, qui déclare une liste de type `it.resto.Vote`, accessible par le champ `votes`.

Pour chaque Event, on garde également une trace de l'utilisateur qui crée l'Event, dans le champ `owner`, qui permet de faire une relation de type 1 ↔ 1 entre un Event et un User.

Terminons avec le Vote qui va relier un Event, un User et la liste des Restaurant que le participant aura sélectionnés.

```
1 class Vote {
2
3     User user
4
5     static hasMany = [restaurants:
6         Restaurant]
7
8     static belongsTo = [event: Event]
9
10    String toString() {
11        def listRestautant
12        for (restaurant in restaurants)
13            {
14                listRestautant += restaurant
15                .name + ' - '
16            }
17        return this.user.login + ' : ' +
18            listRestautant
19    }
20
21    static constraints = {
```

La classe `it.resto.User` définie dans le Vote représente la relation lien direct de type 1 ↔ 1, ce qui

signifie qu'un vote est attaché à un seul et unique User.

Les références vers les restaurants sélectionnés par l'utilisateur sont déclarées avec le mot clef *hasMany*. Un Vote est relié à un ou plusieurs Restaurant.

L'Event est déclaré avec *belongsToMany*, indiquant que le Vote lui appartient. On retrouve son équivalent en *hasMany* dans la classe *it.resto.Event*.

En supprimant un Event, les Vote associés seront également supprimés.

Dans tous les cas, l'identifiant technique *id* est implicite.

Voilà, nous sommes prêts, GORM va faire le reste. Il s'agit d'une surcouche au framework hibernate qui génère automatiquement toute une série de méthodes telles que *load*, *save*, *exists*... mais également une partie plus « magique » ; des méthodes de recherches comme *findBy\** qui sont définies en fonction de la classe. Celles-ci sont utilisables directement dans la suite du projet (implémentation implicite), ce qui réduit le code de l'application et les tests à réaliser.

Par exemple, pour la classe *it.resto.Restaurant*, on retrouvera la *findByName* ou *findByNameLike*.

Dans le tableau ci-dessous, quelques exemples de suffixe pour le *findBy*.

<b>findBy...</b>	
InList	Dans une liste de valeurs données
LessThan	Inférieur à une valeur donnée
LessThanEquals	Inférieur ou égal à une valeur donnée
GreaterThan	Supérieur à une valeur donnée
GreaterThanEquals	Supérieur ou égal à une valeur donnée
Like	Recherche d'un pattern dans une chaîne
Ilike	Similaire au Like, mais insensible à la casse
NotEqual	Différent de la valeur donnée
InRange	Se situant entre deux valeurs ordonnées
Rlike	Similaire au Like, mais avec la possibilité de réaliser des expressions régulières
Between	Se situant entre deux valeurs
IsNotNull	Contenu n'est pas null
IsNull	Contenu est null

## 5 Mise en place des premiers tests

En parallèle de la création des différentes classes de domain (quatre fichiers dans *it-resto/grails-app/domain/it/resto*), Grails a également généré des classes de test (dans *it-resto/grails-app/test/unit/it/resto*). Elles se présentent toutes de la même manière avec une méthode *setup*, qui sera exécutée avant chaque méthode de test, ainsi qu'une méthode *cleanup* exécutée après le test.

Il ne reste plus qu'à écrire les méthodes de test pour valider le bon fonctionnement des classes du domain.

Ces méthodes doivent être écrites en plusieurs sections, connues sous le nom de given-when-then :

- la section *given* décrit l'état du système avant de débiter le scénario. Ce sont les conditions préalables ;
- la section *when* est le déroulement du scénario à spécifier ;
- la section *then* va décrire et tester les changements attendus suite au scénario réalisé dans la section *when*.

Un test sur la création et la validation du domain Restaurant ressemblera à ce qui suit. Dans ce premier test, nous allons vérifier que les contraintes sur

le nom du restaurant sont bien respectées (ni null, ni blank).

```

1 void 'test constraint Restaurant'() {
2     given:
3         mockForConstraintsTests
4             Restaurant
5
6     when: 'the restaurant name is
7         null'
8
9     def restaurant = new Restaurant
10        ()
11        restaurant.name = null
12
13    then: 'validation should fail'
14        !restaurant.validate()
15        restaurant.hasErrors()
16        print restaurant.errors['name']
17
18    when: 'the restaurant name is
19        blank'
20        restaurant.name = ''
21
22    then: 'validation should fail'
23        !restaurant.validate()
24        restaurant.hasErrors()
25        print restaurant.errors['name']
26
27 }
```

Dans la section *given*, la seule chose définie est le *mockForConstraintsTests* qui « mocke » la classe

*it.resto.Restaurant* (toute la machine Grails n'est pas lancée durant les tests). Ce « mock » ajoute la méthode *valide()* à la classe, mais permet aussi de détecter plus simplement des erreurs avec la propriété *errors*. De plus, les messages sont « allégés » et plus simples à analyser.

Par contre, il y a une limitation. Il ne faut pas oublier que le contexte d'exécution est un test unitaire et non pas d'intégration. Il n'y a pas d'ajout de méthodes au runtime par GORM par exemple (*findBy\**, etc).

Dans la section *when*, c'est la zone d'exécution du scénario de test. Ici, c'est un objet *it.resto.Restaurant* qui est créé, mais le nom (*name*) est null (explicitement déclaré).

Dans la section *then*, on valide l'état de sortie

du scénario. La méthode *valide()* va déterminer si toutes les contraintes sont respectées.

Toutes les instructions de la section *then* doivent être vraies, le contraire indiquant une erreur du test. Pour simplifier un peu plus la détection des erreurs, la méthode *hasErrors()* indique si des erreurs ont été rencontrées, puis le message est disponible en fonction du champ en erreur. Dans cet exemple, ce sera *restaurant.errors['name']*.

Pour démarrer le test, lancez la commande *test-app* (ou *grails test-app* si vous avez quitté le mode interactif). À la fin, un rapport est disponible dans `\target\test-reports\`.

Il est recommandé de réaliser cette étape au fur et à mesure de la création des classes du domaine.

## 6 Peuplement de la base

Le squelette de l'application a été réalisé et le modèle de données est prêt (définition des objets domaine ainsi que les tests unitaires sur les contraintes). Avant de réaliser les premiers écrans et pour avoir des éléments à y afficher, il faut peupler la base de données.

Cela se fait simplement, depuis le fichier *grails-app/conf/BootStrap.groovy* qui contient la méthode *init* exécutée au lancement de l'application et la méthode *destroy* quand l'application est arrêtée.

Il suffit de mettre en place le code suivant dans la méthode *init* :

```
1 if (!Restaurant.count()) {
2     print 'Create'
3     new Restaurant(name: 'King Croissant')
      .save(failOnError: true)
4     new Restaurant(name: 'Les trois mi-
      temps').save(failOnError: true)
5 }
```

*Restaurant.count()* dénombre les restaurants en base. Cette condition évite de créer systématiquement les mêmes éléments à chaque lancement de l'application (ce qui arrive régulièrement en phase de développement).

Pour créer un nouveau Restaurant et le stocker en base, on le fera de cette façon :

```
1 new Restaurant(name: 'King Croissant').
  save(failOnError: true)
```

La même chose doit être faite pour les autres éléments du modèle : User, Event et Vote.

Pour la base de données sous-jacente, Grails propose une connexion à la base H2 en natif, en mode stockage en mémoire et une console d'administration accessible via [lien 34](#). L'identifiant par défaut est admin et il n'y a pas de mot de passe.

## 7 Conclusion

Voilà, nous venons de voir comment :

Attention ! Si vous utilisez cette base pour votre application de production, n'oubliez pas de changer le mot de passe admin et de bloquer l'accès à la console, sinon cela revient à exposer tout le contenu de la base sur Internet.

La configuration de la base se fait dans le fichier *grails-app/conf/DataSource.groovy*, au niveau de la propriété *environments.development.datasource* :

```
1 dataSource {
2     dbCreate = "create-drop"
3     url = "jdbc:h2:mem:devDb;MVCC=TRUE;
      LOCK_TIMEOUT=10000"
4 }
```

Initialement, la propriété *dbCreate* est à *create-drop*, mais elle peut prendre d'autres valeurs, selon le besoin :

- *create* : supprime les tables, les index, etc. avant de recréer le schéma au démarrage ;
- *create-drop* : semblable à *create*, mais en supprimant les tables à l'arrêt de l'application ;
- *update* : met à jour la table en créant les tables et les index manquants, sans perdre les données existantes ;
- *validate* : ne fait aucun changement dans la base de données, mais compare le schéma existant avec le schéma configuré dans l'application (objet domaine) et génère un rapport.

**Attention** : *create-drop* et *create* sont à manier avec précaution en dehors d'un environnement de développement.

On retrouve des configurations équivalentes pour les environnements de test et de production.

- créer une application Grails à partir de zéro ;

- mettre en place le modèle de données et poser des contraintes sur celui-ci ;
- réaliser des tests de validation du modèle ;
- peupler la base en vue des premiers développements.

Dans la suite de cet article, nous réaliserons les différents écrans de l'application et irons jusqu'au

déploiement complet de l'application it-resto.

Pour aller plus loin, toutes les sources de l'article sont sur GitHub : [lien 35](#)

La documentation de Grails, très complète, est disponible ici : [lien 36](#)

Enfin, pour ceux qui sont allergiques aux lignes de commandes, il existe le GTS qui permet de simplifier l'approche de l'environnement Grails.

Retrouvez l'article de **Rémi Masson** en ligne : [lien 37](#)



# Libres & Open Source

## Les derniers tutoriels et articles

# Savoir créer un bon mot de passe avec un niveau de sécurité suffisant - Facile à retenir mais difficile à deviner

Les divers faits d'actualité démontrent qu'il est nécessaire de bien définir son mot de passe. Avec l'ère de la diversité des paramètres de connexion à des réseaux sociaux, des forums, des blogs, etc., il est souvent facile de prendre le même mot de passe pour tous... Mais est-ce la bonne solution!!!

## 1 Définition

La définition d'un mot de passe est une séquence de caractères, de mots, de chiffres ou d'un mélange de tout, vous permettant ainsi de vous identifier pour accéder à un service.



Si vous êtes intéressés à la sécurité de vos données, la suite de ce tutoriel va vous servir.

Je ne vous donnerai pas la recette « miracle », mais juste des pistes pour améliorer la robustesse de vos mots de passe.

## 2 Comment choisir un bon mot de passe ?

Les études que vous pouvez voir sur Internet démontrent que bien souvent les mots de passe sont vulnérables, pour preuve, ceux qui ressortent régulièrement dans des sondages sont :

- 123456 ;
- 000000 ;
- password ;
- 12345678 ;
- qwerty ;
- azerty ;

- abc123 ;
- 111111 ;
- etc.

### 2.1 Le nombre de caractères

C'est un mot qui doit contenir un certain nombre de caractères et de différents types : lettres minuscules et majuscules, chiffres et caractères spéciaux.

Un bon mot de passe doit contenir au moins **huit caractères**. Comme le démontre cette synthèse :

Mot de passe	Possibilités	Temps de découverte
<i>Uniquement des minuscules</i>		
6 caractères	308 915 776	0 sec
9 caractères	5 429 503 678 976	9 min
<i>Avec des minuscules et des majuscules</i>		
6 caractères	19 770 609 664	1 sec
9 caractères	2 779 905 883 635 712	3 jours
<i>Avec des minuscules, des majuscules et des chiffres</i>		
6 caractères	56 800 235 584	5 sec
9 caractères	13 537 086 546 263 552	15 jours
<i>Avec des minuscules, des majuscules, des chiffres et des caractères spéciaux</i>		
6 caractères	2 699 554 153 024	4 min
9 caractères	4 435 453 859 151 328 768	14 années



Certains diront, il suffit de mettre un mot de passe de 15 ou 20 caractères et vous serez tranquille !

« prénom » et une « année de naissance », et ces programmes le savent !

Le problème est autre, les programmes qui servent à trouver les mots passe fonctionnent différemment. Car il vous est souvent demandé de mettre des lettres et des chiffres. Il est si facile de saisir un

## 2.2 L'utilisation

Il est recommandé et vivement conseillé de n'utiliser un mot de passe qu'une fois. Car si l'un de vos comptes est piraté, cela n'en rendra que plus facile de pirater les autres.

## 3 Définir un mot de passe sûr

Pour que le mot de passe soit sûr, il va falloir passer par différentes étapes.

### 3.1 Étape 1

La première va consister à définir une *trame* pour le mot de passe.

Il existe pour cela différentes façons de faire, les plus connues sont :

- la méthode phonétique qui consiste à trouver une phrase facile à retenir et de se servir des sons des syllabes pour faire le mot de passe : « J'ai acheté 8 CD pour 100 euros cet après-midi » s'écrit « ght8cd ?am » ;
- la méthode des premières lettres qui consiste à trouver une phrase facile à retenir et de prendre la première lettre de chaque mot pour faire le mot de passe : « un tiens vaut mieux que deux tu l'auras » s'écrit « 1tvmq2tl'a ».

### 3.2 Étape 2

À ce niveau, nous obtenons un mot constitué de lettres minuscules. Il va donc falloir rajouter des majuscules, des chiffres et des caractères spéciaux.

Cette étape est la plus complexe, car elle va dépendre de votre personnalité, et c'est ce qui va en faire votre mot de passe, et il sera ainsi unique.

### 3.2.a Voici quelques cas possibles

#### 3.2.a.a Cas 1

Vous pouvez créer un code, en définissant certains caractères par d'autres, par exemple :

- c par € ;
- d par 1 ;
- v par Z ;
- etc.

Inutile de faire tout l'alphabet, quelques caractères suffiront. Vous pourrez ainsi introduire des chiffres et des caractères spéciaux.

Par exemple, si vous prenez les paroles d'une chanson et avec les caractères cités ci-dessus : « Je vous parle d'un temps Que les moins de vingt ans Ne peuvent pas connaître » s'écrit « jZp1'utqlm1Zanpp€ ».

#### 3.2.a.b Cas 2

Vous pouvez définir une fréquence sur les majuscules, par exemple tous les trois caractères.

Et en faire de même avec les chiffres, où vous choisissez un nombre d'au moins quatre chiffres. Ensuite, il vous suffit de l'inclure tous les deux caractères ou d'en mettre un au début, un à la fin et deux au milieu.

Par exemple, si vous prenez les paroles d'une chanson : « Je vous parle d'un temps Que les moins

de vingt ans Ne peuvent pas connaître », en prenant les tous les trois caractères pour la majuscule et l'année 2014, cela donne : « 2jvpD'utq0L1mdvAnppC4 »

### 3.2.a.c Cas 3

Vous pouvez aussi décaler les lettres de l'alphabet d'un nombre de caractères ou en faire de même avec les touches du clavier.

Par exemple, si vous prenez les paroles d'une chanson : « Je vous parle d'un temps Que les moins de vingt ans Ne peuvent pas connaître » avec le clavier en décalant d'une case s'écrit « kb^(iysmùfbz,^^ »

### 3.2.b Conclusion

Il existe beaucoup d'autres méthodes, et vous pouvez créer la vôtre. **Le but est qu'une fois que vous êtes arrivé à cette étape vous obteniez un mot de passe personnalisé et dont vous pouvez vous souvenir facilement.**

## 4 Générateur de mots de passe

Il existe une autre façon de créer des mots de passe qui est de passer par des générateurs de mots de passe.

En voici quelques-uns en ligne :

- site 1 : [lien 38](#) ;
- site 2 : [lien 39](#) ;

## 5 Conclusion

Nous venons de voir comment vous pouvez définir un mot de passe. Mais un mot de passe ne sera jamais sûr, c'est pourquoi il est important de le modifier régulièrement.

**La fréquence du changement dépend principalement des données contenues, mais il est préconisé de le faire tous les trois mois, et pour les systèmes contenant des informations bancaires, il est conseillé de le faire plus souvent.**

**Il est aussi bien important de ne pas le noter sur un bout de papier qui pourrait finir dans de « mauvaises mains »...**

**Il en est de même, si vous utilisez vos pa-**

Retrouvez l'article de *Vincent Viale* en ligne : [lien 44](#)

### 3.3 L'unicité

Il existe des méthodes pour créer des mots de passe uniques à partir du mot de passe défini dans l'étape précédente.

Pour cela, il vous suffit de rajouter des informations à la fin, au début ou au milieu de votre mot de passe, en prenant l'exemple du cas 1 « jZp1'utqlm1Zanpp€ » , cela donne :

- de prendre la première et la dernière lettre du système (Facebook en « fk », Orange en « oe », etc.) : ce qui peut donner « jZp1'utqlm1Zanpp€\_fk » ;
- de prendre deux derniers caractères du système (Facebook en « ok », Orange en « ge », etc.) : ce qui peut donner « jZp1'utqlm1Zanpp€\_ok » ;
- de prendre certains caractères et de les insérer à la suite d'un caractère spécial : ce qui peut donner « jZp1'fkutqlm1Zanpp » ;
- etc.



Vous pouvez aussi rajouter des caractères spéciaux pour introduire le système.

- site 3 : [lien 40](#) ;
- etc.

L'inconvénient est que si vous passez par ces sites, il vous faudra trouver une méthode pour retrouver votre mot de passe ou sinon, il ne vous reste plus qu'à l'apprendre par cœur.

**ramètres sur un ordinateur public, de ne pas oublier de vous déconnecter à la fin.**

Maintenant vous pouvez trouver sur Internet des sites qui vous permettront d'en tester la robustesse. Le résultat peut varier d'un site à l'autre, en voici quelques-uns :

- site 1 : [lien 41](#) ;
- site 2 : [lien 42](#) ;
- site 3 : [lien 43](#) ;

Vous avez maintenant en main certains éléments qui vous permettront de créer des mots de passe corrects.

# Emmabuntüs - Premiers pas sur la voie Libre ;)

Le but de ce manuel est de fournir les éléments de base à tout nouvel acquéreur d'une machine sous Emmabuntüs pour lui permettre de bien débiter.

## 1 Le but de ce manuel

Le but de ce manuel est de fournir les éléments de base à tout nouvel acquéreur d'une machine sous

Emmabuntüs pour lui permettre de bien débiter.

## 2 Distribution : Emmabuntüs

Les distributions GNU (lien 45)/Linux (lien 46) Emmabuntüs ont été conçues pour faciliter le reconditionnement des ordinateurs donnés aux associations humanitaires, en particulier aux communautés Emmaüs (d'où son nom) (lien 47), et favoriser la découverte de GNU (/)Linux (lien 49) par les débutants. Un troisième objectif est de prolonger la durée de vie du matériel pour limiter le gaspillage entraîné par la surconsommation de matières

premières (pour de plus amples informations, voir info.emmabuntus.org : lien 50).

- Page principale : lien 51 ;
- Forum : lien 52 ;
- Wikipédia : lien 53 ;
- Articles sur Emmabuntüs dans la blogosphère : lien 54 ;
- Revues de presse : lien 55.

## 3 Partie matérielle

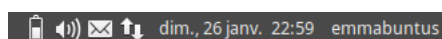
### 3.1 Se connecter par câble Ethernet (réseau filaire)

Branchez simplement votre câble Ethernet à votre machine : la connexion s'établit de façon automatique, et vous devez voir s'activer l'icône des connexions (deux flèches haut/bas) dans la zone de notification en haut à droite de votre écran.

Connexion inactive



Connexion active



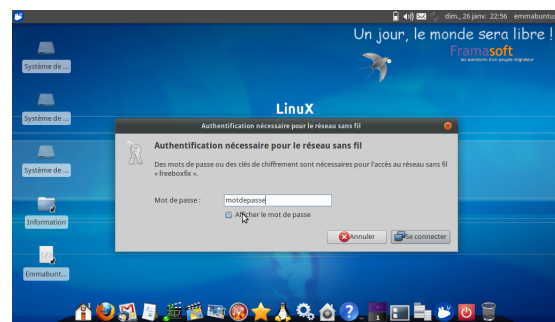
### 3.2 Se connecter en Wi-Fi (réseau sans fil)

Cliquez sur l'icône des connexions (deux flèches). Voir images ci-dessus dans la zone de notification en haut à droite de votre écran.



Puis sélectionnez votre réseau Wi-Fi qui doit apparaître, sinon cliquez sur la ligne « Plus de réseaux ».

Renseignez le code de sécurité de votre réseau Wi-Fi, et, pour ne pas faire d'erreur, pensez à cocher la case « Afficher le mot de passe ».





### 3.3 Clé Wi-Fi reconnue automatiquement

Si votre PC n'a pas de carte Wi-Fi incorporée, nous vous conseillons d'utiliser l'Adaptateur USB Wireless-N 150 WNA1100 de chez Netgear (lien 56),



à ne pas confondre avec le modèle USB Wireless N150 Nano WNA1000M **non compatible** avec Emmabuntüs (lien 57).

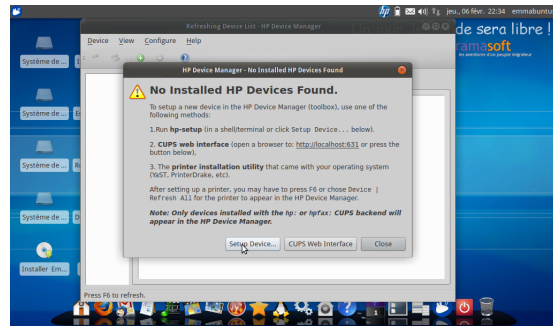


Certaines clés Wi-Fi ne sont pas reconnues automatiquement, c'est la raison pour laquelle nous vous conseillons la clé WNA1100 toujours en vente à environ 15 €. Mais il est possible de faire fonctionner des clés non reconnues automatiquement, comme la WNA1000M, et d'autres : voir le chapitre clés Wi-Fi sur le Wiki d'Ubuntu-fr (doc.ubuntu-fr.org/Wi-Fi : lien 58).

### 3.4 Ajout d'une imprimante

Tous les modèles d'imprimantes ne sont pas compatibles avec Linux et il faudra peut-être installer des pilotes additionnels non libres (par exemple Epson). La procédure consiste à brancher l'imprimante et la faire détecter par le système qui proposera les drivers adaptés.

Si l'imprimante n'est pas détectée automatiquement par l'ordinateur et que votre imprimante est de marque HP, lancez l'utilitaire HPLip (lien 59) inclus dans Emmabuntüs sous « Utilitaire », cliquez sur l'icône « HPLIP Toolbox », puis sur « Setup Device », et renseignez les champs de cette interface.



Cet utilitaire est disponible uniquement en anglais, et pour plus de renseignements voir le Wiki d'Ubuntu-fr (doc.ubuntu-fr.org/hplip : lien 60).

Si l'imprimante est d'une autre marque, voir le tableau de compatibilité sur la page dédiée du Wiki d'Ubuntu-fr : (doc.ubuntu-fr.org/imprimante : lien 61).

### 3.5 Ajout d'un scanner

Branchez le scanner et lancez l'outil SimpleScan (lien 62) dans le menu Photo. Lancez la numérisation et enregistrez sous le format souhaité (format à choisir en bas à gauche) .pdf est le plus utilisé pour les documents, .jpg pour les images. Si vous avez des soucis de fonctionnement, nous vous conseillons de lire cette page dédiée aux scanners sur le Wiki d'Ubuntu-fr : (doc.ubuntu-fr.org/scanner : lien 63).

### 3.6 Ajout d'une clé 3G

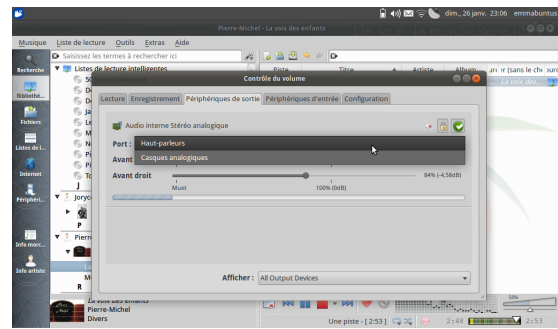
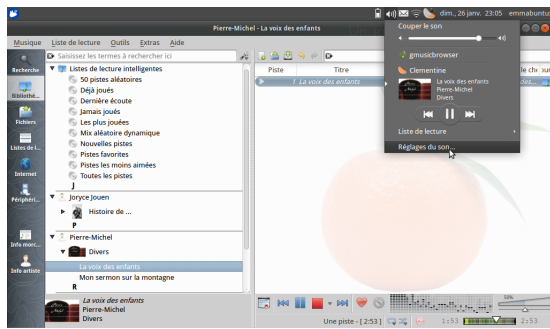
La configuration d'une clé 3G est une opération assez compliquée, voir la liste des clés 3G sur le Wiki d'Ubuntu (lien 64), ou sur les pages dédiées aux clés 3G Sfr (lien 65), et Orange (lien 66).

Pour contourner ce problème, nous vous conseillons de vous servir de votre téléphone portable 3G comme routeur, en faisant une connexion avec votre ordinateur en Wi-Fi ou par câble USB. Voir comment connecter votre téléphone à un ordinateur ou faire un pont sur votre téléphone en faisant une recherche par exemple avec Android version \_Android Ubuntu.

### 3.7 Réglage audio

Les lecteurs Clementine (lien 67) et Rhythmbox (lien 68) contiennent des morceaux de musique, afin de pouvoir tester directement le son sur l'ordinateur.

Lancez par exemple le lecteur Clementine. Pour cela, cliquez dans la catégorie « Audio », puis sur « Clementine ». Ensuite, double-cliquez sur un morceau de musique, et si vous n'entendez pas de musique, cliquez sur l'icône représentant un haut-parleur en haut à droite de l'écran, puis sur « Réglages du son... » :



Puis sous la fenêtre de gestion du contrôle de volume, allez sous l'onglet « Périphériques de sortie », et dans le menu déroulant essayez les différents ports de sortie pour le son.

## 4 Partie logicielle

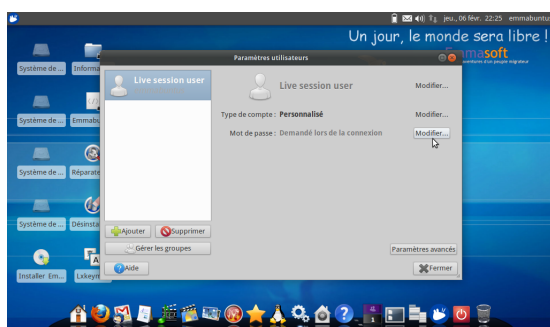
### 4.1 Mot de passe

Le mot de passe par défaut est renseigné sur votre fiche de vente, si ce n'est pas le cas veuillez prendre contact avec votre point de vente.

Ce mot de passe vous sera nécessaire pour effectuer les mises à jour logicielles de votre machine, ou pour installer de nouveaux programmes (toutes tâches d'administration).

### 4.2 Changer son mot de passe

Cliquez sur la catégorie « Utilitaire », puis sur l'icône « Users and Groups ». Dans cette nouvelle fenêtre, cliquez sur le bouton « Modifier... », puis renseignez l'ancien mot de passe et saisissez deux fois le nouveau mot de passe.



### 4.3 Réinitialiser son mot de passe

La réinitialisation du mot de passe est nécessaire si vous l'avez oublié. Pour cela, lors du démarrage de la machine, maintenez enfoncée la touche « **Shift** » à droite du clavier (Touche majuscule droite)

### 3.8 Test de votre webcam

Lancez le logiciel Cheese ([lien 69](#)) afin de vérifier que votre webcam est bien fonctionnelle, pour cela cliquez sur la catégorie « Vidéo », puis sur « Cheese », et vous devriez voir l'image produite par votre webcam (votre bobine généralement !). Si cela ne fonctionne pas voir sur le Wiki d'Ubuntu-fr : [doc.ubuntu-fr.org/webcam \(lien 70\)](#)

jusqu'à l'apparition du menu de lancement du système sur fond noir. Ensuite, sélectionnez la 2e ligne du menu de lancement contenant le texte suivant : « Linux (mode de dépannage) », puis tapez sur la touche Entrée pour valider cette ligne. Le système va se lancer et vous allez arriver dans un nouveau menu sur fond bleu, alors veuillez sélectionner la ligne « Passage en mode Root », puis dans cette nouvelle fenêtre de console sur fond noir, veuillez taper la ligne suivante : `mount -rw -o remount /`

Puis validez, et ensuite tapez cette ligne : `passwd <votre identifiant>`, par exemple sur les machines vendues dans nos points de vente, il faut taper : `passwd emmabuntus`, puis validez. Vous devez ensuite renseigner deux fois le nouveau mot de passe.

Si tout se passe correctement le message suivant devrait apparaître :

`passwd : password updated successfully .`

Puis tapez cette commande pour redémarrer la machine : `reboot`.

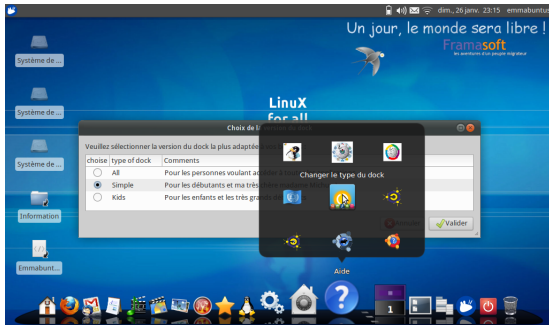
### 4.4 Dock

Le Dock ou lanceur d'applications est la pierre angulaire d'Emmabuntüs, et lui apporte indépendance et originalité par rapport aux distributions Ubuntu dont elle est issue. C'est Cairo-Dock ([lien 71](#)) qui est utilisé depuis la première version d'Emmabuntüs.

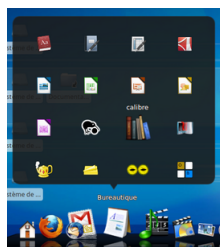


Pour changer de niveau d'utilisateur du Dock, cliquez sur « Aide », puis « Changer le type du dock

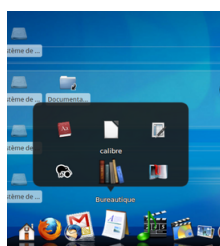
», et sélectionnez le niveau désiré, voir l'image ci-dessous :



Ce Dock se décline en trois niveaux d'utilisation : *All*, *Simple*, et *Kids*. Il est escamotable ou pas en fonction du format de l'écran. Exemple des trois niveaux de Dock pour la catégorie bureautique :



Niveau expert



Niveau débutant



Niveau enfant

Le lancement du Dock n'est pas obligatoire. Pour l'arrêter, il faut modifier les lignes ci-dessous contenues dans le fichier `/.profile` en lançant par exemple la commande : `geany /.profile`



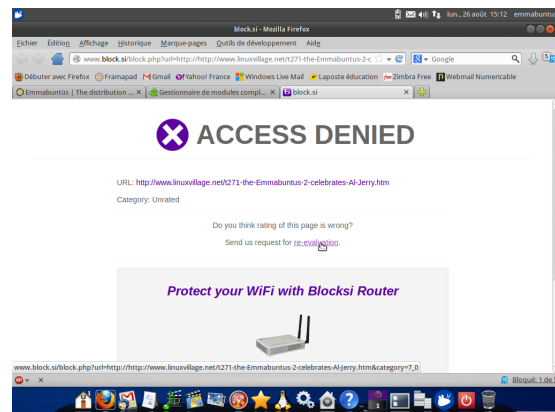
```

1 # Config pour le projet
  Emmabuntus
2 #Set Dock_xx to 1 to
  activate the Cairo-Dock
  on XFCE or LXDE or
  OpenBox
3
4 export Dock_XFCE=< 1 >
5 export Dock_LXDE=< 0 >
6 export Dock_OpenBox=< 0 >
  
```

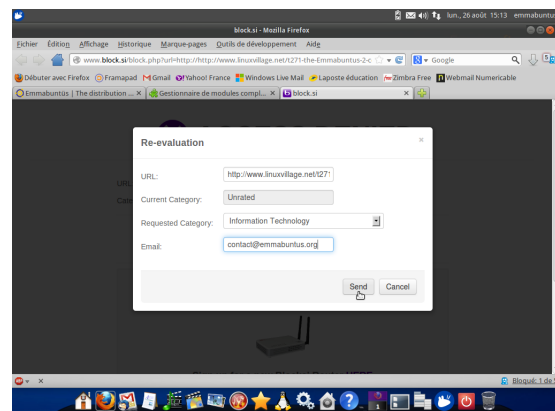
#### 4.5 Blocage de certains sites Internet

Emmabuntus contient deux navigateurs Internet Firefox (lien 72) et Chromium (lien 73) qui incluent des extensions pour la protection des mineurs (Block.si), mais aussi contre la publicité (lien 74) et les tentatives de Phishing (ou Fishing) (lien 75).

L'extension de protection des mineurs peut aussi bloquer l'accès à certains sites, voir l'image ci-dessous du blocage d'un site sans catégorie :

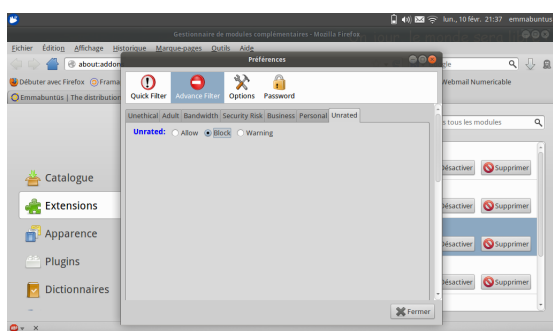
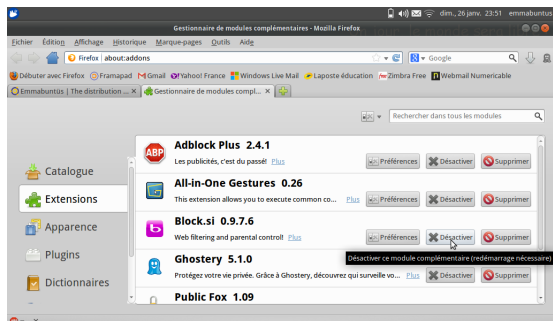


Pour ne plus avoir ce blocage avec les sites sans catégorie, il faut demander la réévaluation du site, simplement en renseignant la catégorie, et en communiquant son adresse mail :



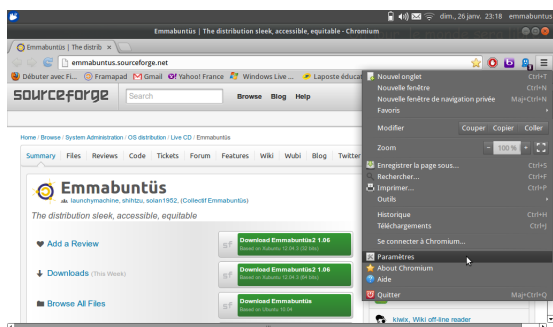
Il est également possible de désactiver cette protection.

**Pour Firefox :** cliquez sur « Outils de développement », puis sur « Modules complémentaires », et sous l'onglet « Extensions », cliquez sur le bouton « Désactiver » de l'extension Block.si.

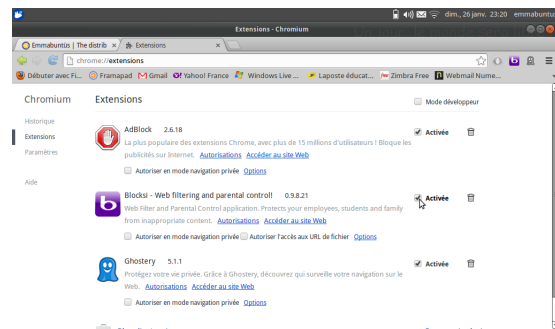


Il est également possible de désactiver certains paramètres de filtrage du contrôle parental, pour cela cliquez sur le bouton « Préférences » de l'extension Block.si, puis sélectionnez l'onglet « Advance Filter », puis choisissez les types de sites que vous voulez autoriser (Allow) ou bloquer (Block). Le filtre « Unrated » est celui qui pose le plus de problèmes, car il englobe les sites n'appartenant à aucune catégorie, et de ce fait permet une protection maximale, c'est pour cette raison que nous l'avons activé, mais vous êtes libre de le désactiver.

**Pour Chromium :** cliquez sur le symbole carré avec trois barres horizontales à droite du navigateur, puis sur « Paramètres ».



Sous l'onglet « Extensions », cliquez sur désactiver Block.si.

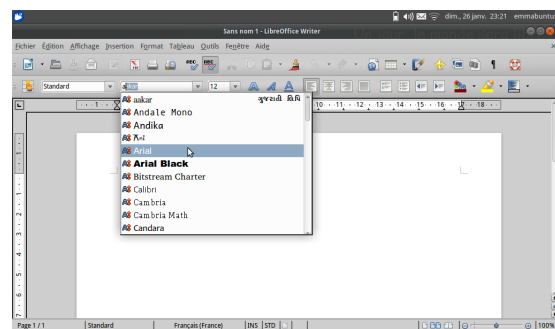


Plus d'informations sur la configuration du contrôle parental sous Emmabuntüs 2 sur le site de Comment Ça Marche : lien 76.

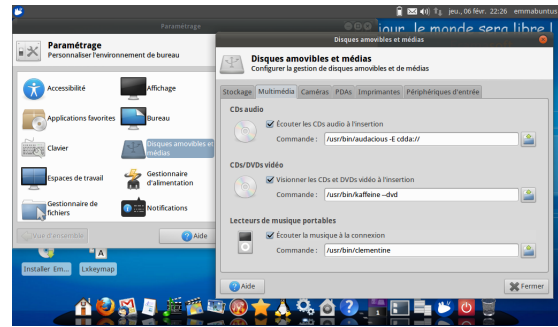
#### 4.6 Compatibilité avec la suite Microsoft Office


La catégorie bureautique a particulièrement été étudiée afin de fournir des outils adaptés à chaque public avec trois suites bureautiques : LibreOffice pour les utilisateurs avancés (lien 77); AbiWord (lien 78)/Gnumeric (lien 79) pour les débutants, ou pour les machines ayant peu de ressources; et OOo4Kids (lien 80) pour les enfants.


Emmabuntüs est prévue pour installer facilement les polices de caractères non libres de la famille Arial (Microsoft Office 2003) (lien 81), et de la famille Calibri (Microsoft Office 2007/2010) (lien 82). Pour vérifier que ces polices de caractères sont présentes dans votre ordinateur, ouvrez l'application Writer de LibreOffice (lien 83), et dans le sélecteur de polices tapez la lettre A pour vérifier que la police Arial est présente. De la même façon, rouvrez le sélecteur et tapez la lettre C afin de vérifier que la police Calibri est présente.



Si l'une ou l'autre des polices n'est pas présente, cliquez sur « Maintenance », puis « Logiciels Non libres », et cliquez sur l'une des icônes contenant une lettre T pour installer la police non présente sur votre machine. Renseignez le mot de passe quand il est demandé.




 Pour pouvoir installer ces polices de caractères, votre ordinateur doit être raccordé à Internet.

 Par défaut LibreOffice enregistre les documents dans un format ouvert appelé ODT (Open Document Text) (lien 84), ce format n'étant pas lisible par défaut dans les anciennes versions de la suite bureautique Microsoft Office (lien 85), il est judicieux de faire aussi une copie au format .doc version Microsoft Word (lien 86) 97/2000/XP/2003 de votre document original pour l'échanger avec des personnes qui n'utilisent pas LibreOffice, et aussi de joindre la version PDF.

#### 4.7 Lire un CD/DVD

Il suffit de mettre un CD ou un DVD dans le lecteur, et au bout de quelques secondes le lecteur Audacious (lien 87) s'ouvre pour la lecture des CD Audio, ou Kaffeine (lien 88) pour les DVD.

 Par défaut nous avons configuré Emmentabuntüs pour lancer les médias avec les logiciels ci-dessus, mais si vous préférez utiliser par exemple VLC (lien 89) à la place de Kaffeine, voici la marche à suivre.  
Cliquez sur l'icône « Paramétrage » sous « Utilitaires » dans le dock, puis dans la nouvelle fenêtre sur « Disques amovibles et médias », et sur l'onglet « Multimédia ».  
Remplacez la ligne concernant le lecteur Kaffeine par la ligne suivante :  
`/usr/bin/vlc -playlist-autostart dvd ://`

#### 4.8 Voir des vidéos sur Internet

Afin de pouvoir regarder des vidéos sur Internet, par exemple sur YouTube (lien 90), nous avons été obligés d'intégrer le lecteur Adobe Flash (lien 91), qui est un lecteur non libre (qui peut donc contenir des espions logiciels) (lien 92), et dont le format n'est pas documenté, ce qui empêche l'élaboration d'une version libre, préservant la vie privée des utilisateurs.

#### 4.9 Visioconférence sur Internet

Afin de pouvoir effectuer des vidéoconférences sur Internet facilement, nous avons été obligés d'intégrer l'application Skype (lien 93), qui est non libre (qui peut donc contenir des espions logiciels) (lien 94). D'autres solutions libres sont disponibles dans Emmentabuntüs comme Empathy (lien 95) ou Ekiga (lien 96), par contre vous serez obligé de créer un nouveau compte et vos amis seront aussi obligés d'utiliser une application compatible autre que Skype, car ce dernier n'étant pas libre, il est impossible de pouvoir développer un logiciel libre compatible.

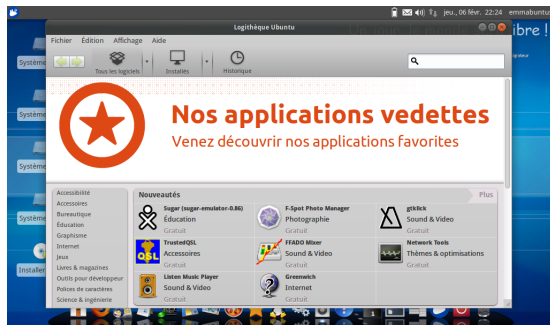
#### 4.10 Ajout de logiciels fonctionnant sur Linux

Le système Emmentabuntüs (comme tout système Linux) fonctionne avec des « paquets ». Un paquet (lien 97) contient tous les fichiers nécessaires au fonctionnement d'un programme. Pour installer un *paquet*, il existe plusieurs méthodes :

- la logithèque Ubuntu (pour les débutants) : lien 98 ;
- L'interface graphique de gestion de *paquets* Synaptic (pour les utilisateurs avancés) : lien 99 ;
- La ligne de commande apt-get de gestion de *paquets* (pour les experts) : lien 100.

Pour les débutants l'installation de nouveaux logiciels a été simplifiée grâce à l'utilisation de la logithèque incluse dans le système. Pour cela, cliquez sur « Maintenance », puis « Logithèque » (icône représentant un sac). Ensuite, faire une recherche avec le nom du programme, ou en utilisant la navigation par catégories.

Si le logiciel n'est pas présent, il faut faire une recherche sur Internet en utilisant : « le nom de votre programme » associé au mot Ubuntu, sinon voir la documentation sur le Wiki d'Ubuntu-fr (doc.ubuntu-fr.org : lien 101).



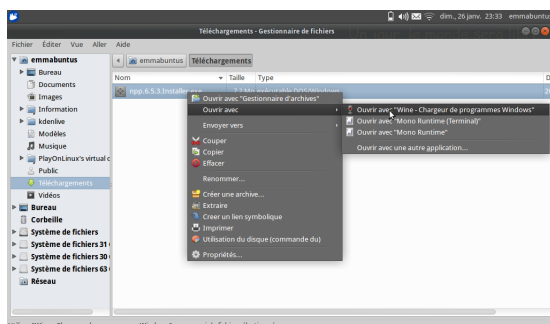
#### 4.11 Ajout de logiciels fonctionnant sur Windows

Afin de permettre une plus grande compatibilité avec des programmes ne fonctionnant que sur les plates-formes Windows, WINE (lien 102) est intégré de base dans Emmabuntüs. Il permet par exemple de faire fonctionner l'excellent éditeur de texte Notepad++ : lien 103.

Tous les programmes Windows ne peuvent pas fonctionner sous Linux même en utilisant WINE, voir la liste des applications compatibles : lien 104

Nous vous invitons plutôt à rechercher un logiciel Libre Linux vous permettant d'obtenir un résultat équivalent et peut-être même d'enrichir les auteurs de ce logiciel de vos améliorations.

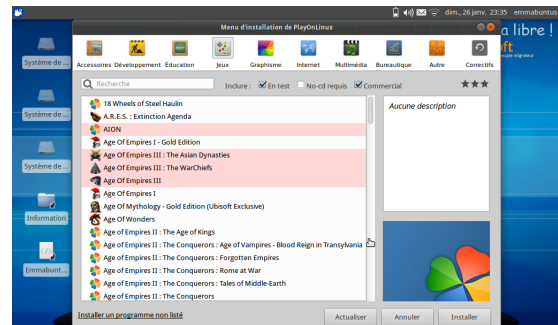
Pour installer une application Windows, téléchargez le programme exécutable (.exe), puis faites un clic droit sur cet exécutable, et sélectionnez « Ouvrir avec : Wine - Chargeur de programmes Windows » :



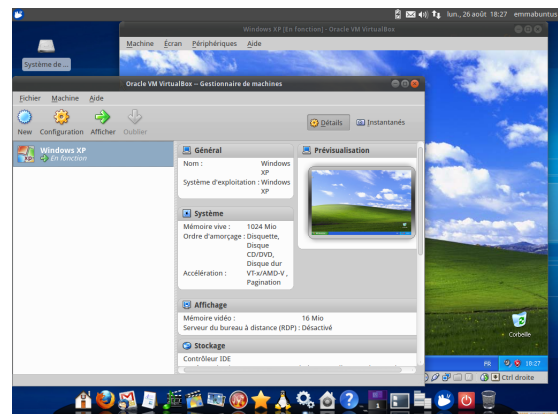
Emmabuntüs contient aussi PlayOnLinux (lien 105), application permettant de faire fonctionner certains logiciels développés pour la plate-forme Windows, et en particulier dans le domaine du jeu vidéo, mais pour cela il faut posséder les CD ou DVD originaux de ces jeux.

Pour utiliser cette application, cliquez dans la catégorie « loisirs », puis sur « PlayOnLinux », et suivre le guide pour installer votre application.

N'oubliez pas que certains jeux nécessitent des ordinateurs plus puissants que celui que vous possédez, veuillez regarder les prérequis de votre jeu.



Toujours afin de permettre une transition entre les plates-formes Windows, et Linux, VirtualBox (lien 106) est disponible pour éventuellement faire fonctionner son ancien système Windows sous Emmabuntüs, et continuer à utiliser des logiciels ne pouvant pas fonctionner sous Linux, et n'ayant pas d'équivalence complète. Voir cette liste d'équivalences Windows Linux (lien 107).



Virtualbox est une machine virtuelle et ne peut en aucun cas remplacer votre ancien système en ce qui concerne les interfaces physiques qui sont des interfaces simulées dans la machine virtuelle. En d'autres termes, il est fort peu probable d'arriver à faire fonctionner votre ancienne imprimante ou scanner à travers VirtualBox.

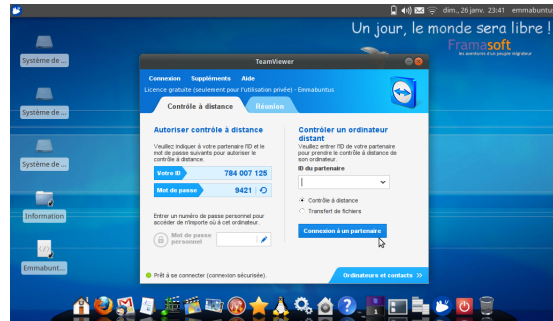
La mise en œuvre de VirtualBox nécessite d'avoir des connaissances en informatique, et de posséder les versions des CD ou DVD d'installation de ces autres systèmes d'exploitation.

#### 4.12 Aide à distance

TeamViewer est un logiciel gratuit, mais non libre permettant le contrôle à distance entre deux machines, afin de pouvoir aider et former les nouveaux utilisateurs. Cette fonctionnalité n'est possible que dans un contexte non commercial : lien 108.

Pour lancer TeamViewer, cliquez sur l'icône « Maintenance », puis sur « TeamViewer » (icône bleue avec une flèche à double sens). Cette opération doit se faire sur les deux machines que l'on souhaite connecter ensemble avec la même version de l'application. La personne qui veut prendre le contrôle de la machine distante demande à son interlocuteur (partenaire) de lui fournir le « Code ID » (Code à neuf chiffres en haut à gauche de l'interface de connexion) et le saisit dans la partie de droite de

l'interface. Si le bon code est saisi, l'application demande de renseigner le mot de passe du partenaire, code composé de six lettres et chiffres, situé sous le code ID du partenaire. La connexion s'établit alors entre la machine maître et la machine du partenaire.

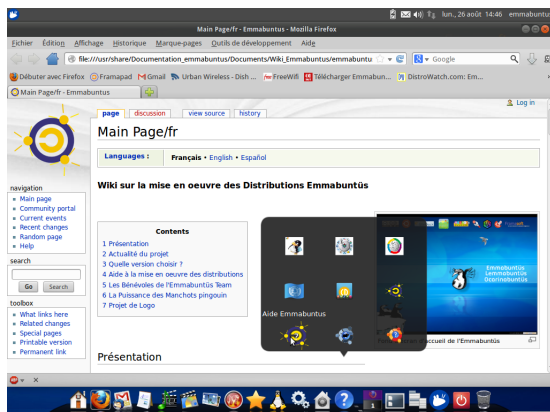


## 5 Aide

Emmabuntüs intègre l'aide hors ligne directement à partir du Dock, mais aussi l'ensemble des tutoriels directement accessible à partir du bureau, cliquez sur l'icône Emmabuntüs Information.

Si vous ne trouvez pas de solution à votre problème, voir :

- F.A.Q : lien 109 ;
- Forum : lien 110.



Retrouvez l'article du *Collectif emmabuntüs* en ligne : lien 111

# ALM



## Les derniers tutoriels et articles

# Guide de Lean Management à l'usage des équipes Agiles

Ce guide gratuit, rédigé par des praticiens expérimentés, vous aidera à démarrer la mise en œuvre de trois techniques lean fondamentales pour améliorer vos pratiques agiles :

- identifier plus finement les développements nécessaires pour résoudre vraiment le problème de vos clients ;
- créer un environnement visuel qui emmène toute votre équipe dans l'amélioration continue ;
- trouver les améliorations qui font la différence, par une démarche structurée de résolution des problèmes.

Chaque pratique est illustrée d'exemples concrets, et accompagnée d'une checklist pour démarrer du bon pied.

Alors qu'attendez-vous ?

## 1 Avant-propos



Les pratiques présentées dans ce guide ouvrent la voie vers une vision différente de l'organisation. Une vision basée sur une conviction.

Le changement vers une organisation à la fois plus efficace et plus respectueuse des personnes est possible. Ce changement naît de la somme des apprentissages individuels.

L'amélioration continue, en définitive, est celle des compétences de chaque collaborateur. L'apprentissage devient

indissociable du travail, et le « qui doit faire quoi pour réussir » devient « qui doit apprendre à faire quoi pour réussir ».

Cette transformation radicale se construit jour après jour, personne par personne, en allant sur le terrain pour aider chaque collaborateur à réussir sa journée. Cela amène chacun à créer plus de valeur pour ses clients, son entreprise et la société, et trouver ainsi un sens à son travail.

Le chemin vers cet idéal a été tracé par nos prédécesseurs pendant plusieurs décennies, et consigné sous la forme de principes et de pratiques qui ont pris le nom de « lean ».

Ce savoir-faire précieux nous a été transmis par Marie-Pia Ignace et Michael Ballé. Nous vous le transmettons à notre tour, en espérant qu'il vous apportera les mêmes moments de satisfaction, les incroyables déclics qui vous feront prendre de la hauteur.

Régis Medina

## 2 Inauguration du pont

Depuis plusieurs années, l'esprit et les pratiques agiles vous ont permis d'améliorer votre satisfaction professionnelle et la performance de vos équipes.

Mais voilà : il reste encore des sources de frustra-



tion. Les autres équipes résistent, les managers ne sponsorisent pas vos initiatives de changement, les clients se plaignent. Il doit bien exister des moyens d'améliorer les choses, mais comment les trouver ?

En vous entraînant aux pratiques lean sélectionnées dans ce livre, vous apprendrez à :

- trouver les leviers de l'amélioration qui amèneront vos équipes à un autre niveau de performance ;
- résoudre les difficultés que vous rencontrez dans vos relations avec d'autres équipes ou le management ;
- livrer des logiciels qui améliorent la vie de vos utilisateurs et dont vous pouvez être fiers.

Ces nouvelles compétences vous apporteront le savoir-faire nécessaire pour insuffler les changements

vitaux au sein des organisations.

Ce livre se structure autour de trois apprentissages fondamentaux. Pour chacun d'entre eux, des praticiens agiles vous racontent comment, en appliquant d'autres pratiques, en adoptant d'autres postures, en s'entraînant à fonctionner différemment, ils ont trouvé des solutions simples à des problèmes qui paraissaient complexes.

Puis des experts décrivent les principes lean mis en œuvre dans les histoires présentées.

Enfin, des préconisations de premiers pas vous guident vers la mise en pratique.

Enfin, des préconisations de premiers pas vous guident vers la mise en pratique.

Nous souhaitons transmettre ce que nous avons appris lors de notre voyage initiatique. Notre promesse : ça en vaut la peine !

### 3 Structure du livre

Ce guide comprend trois chapitres :

1. Comprendre l'attente du client : [lien 112](#) ;
2. Visualiser le challenge et les problèmes : [lien 113](#) ;
3. Trouver les leviers de l'amélioration : [lien 114](#).

Chaque chapitre est organisé de la façon suivante.

- Une description des pratiques agiles sur le thème abordé.

- Trois cas réels d'équipes agiles ayant intégré le management lean dans leurs pratiques. Ils décrivent leur contexte, les exercices qu'ils appliquent, et ce qu'ils y gagnent. À la fin de chaque cas, nous analysons et développons les principes lean mis en œuvre.

- Une description des pratiques lean sur le thème du chapitre.

- Les « premiers pas » que nous proposons de mener pour se lancer.

- Des références de lecture pour aller plus loin.

### 4 Conclusion

Le lean est une pratique. La valeur de ce guide réside dans les « premiers pas », les exercices que nous avons sélectionnés pour vous. Commencez à les mettre en œuvre dès maintenant, pas après pas, et

venez partager vos déclics et vos questions avec les autres praticiens pour que nous progressions tous ensemble.

*Retrouvez l'article de **Philippe Blayo, Antoine Contal, Dominique De Premorel, Pierre Jannez, Christophe Keromen, Régis Medina, Sandrine Olivencia, Christophe Ordano, Raphaël Pierquin et Bruno Thomas** en ligne : [lien 115](#)*

# Liste des liens

## Page 9

**lien 1** : ... <http://wassimchegham.developpez.com/tutoriels/javascript/javascript-pour-les-jedis-episode-1-au-coeur-des-fonctions/>

## Page 11

**lien 2** : ... <http://www.developpez.com/redirect/1428>

**lien 3** : ... <http://www.developpez.com/redirect/1429>

**lien 4** : ... <http://javascript.developpez.com/>

**lien 5** : ... [http://javascript.developpez.com/cours/?page=cote\\_serveur#node](http://javascript.developpez.com/cours/?page=cote_serveur#node)

**lien 6** : ... <https://www.youtube.com/watch?v=USsQRTsWIZo>

**lien 7** : ... <https://github.com/andreareginato/simple-oauth2>

## Page 12

**lien 8** : ... <http://expressjs.com/>

**lien 9** : ... <http://expressjs.com/starter/installing.html>

**lien 10** : ... <https://anywhere-sandbox.idn.laposte.fr/>

## Page 13

**lien 11** : ... <http://www.developpez.com/redirect/1436>

**lien 12** : ... [http://javascript.developpez.com/cours/?page=cote\\_serveur#node](http://javascript.developpez.com/cours/?page=cote_serveur#node)

**lien 13** : ... <https://github.com/BeMyApp-France>

**lien 14** : ... <mailto:equipe-idn.dsic@laposte.fr>

**lien 15** : ... <http://laposte.developpez.com/tutoriels/identite-numerique-la-poste/integration-api-authentification-nodejs/>

## Page 14

**lien 16** : ... <http://www.paulund.co.uk/create-download-icon-css>

**lien 17** : ... <http://www.paulund.co.uk/playground/demo/css-download-icon/>

## Page 15

**lien 18** : ... <http://www.paulund.co.uk/css3-buttons-with-pseudo-classes>

**lien 19** : ... <http://www.paulund.co.uk/playground/demo/css-download-icon/>

**lien 20** : ... <http://www.paulund.co.uk/playground/demo/css-download-icon/>

**lien 21** : ... <http://www.paulund.co.uk/c/tutorials/css-tutorials>

**lien 22** : ... <http://djibril.developpez.com/tutoriels/temp/css-test/>

## Page 16

**lien 23** : ... <http://www.developpez.net/forums/d1420991/environnements-developpement/delphi/langage/variables-globales-section-implementation-objet-singleton/>

## Page 24

**lien 24** : ... <http://jeremy-laurent.developpez.com/tutoriels/delphi/patterns/singleton/>

## Page 25

**lien 25** : ... <http://www.developpez.net/forums/d1454204-2/environnements-developpement/eclipse/eclipse-java/projet-valhalla-incubateur-d-idees-fonctionnalites-preparer-terrain-java-10-a/#post8080534>

## Page 26

**lien 26** : ... <http://www.developpez.com/actu/80483/Un-pour-cent-des-bugs-signalés-ont-été-exploités-Silverlight-et-Apache-de-plus-en-plus-visés-par-les-pirates-selon-le-rapport-de-sécurité-de-Cisco/>

**lien 27** : ... <http://www.developpez.net/forums/d1494706/java/general-java/java-nouvelles-mises-jour-corriger-19-vulnerabilites-desactiver-ssl-3-0-a/>

## Page 27

**lien 28** : ... <http://www.webmotion-framework.org/>

## Page 31

**lien 29** : ... <http://www.webmotion-framework.org/>

**lien 30** : ... <http://julien-ruchaud.developpez.com/tutoriels/introduction-framework-web-webmotion/>

**Page 32**

- lien 31 : ... <http://localhost:8080/it-resto/>
- lien 32 : ... <http://pparrend.developpez.com/tutoriel/mda-intro/>

**Page 33**

- lien 33 : ... <http://ericreboisson.developpez.com/tutoriel/java/groovy/>

**Page 35**

- lien 34 : ... [http://localhost:8080/nom\\_app/console](http://localhost:8080/nom_app/console)

**Page 36**

- lien 35 : ... <https://github.com/masson-r/it-resto>
- lien 36 : ... <http://grails.org/doc/2.4.4/guide/index.html>
- lien 37 : ... [http://remimasson.developpez.com/tutoriels/grails/developper\\_application\\_web/initialisation\\_part1/](http://remimasson.developpez.com/tutoriels/grails/developper_application_web/initialisation_part1/)

**Page 39**

- lien 38 : ... <http://www.generateurdemotdepasse.com/>
- lien 39 : ... <http://www.libellules.ch/passwordgen.php>
- lien 40 : ... <http://www.francelink.net/generateur-de-mot-de-passe>
- lien 41 : ... <https://howsecureismypassword.net/>
- lien 42 : ... <https://pwdtest.bee-secure.lu/>
- lien 43 : ... <http://www.panoptinet.com/panoptipass/>
- lien 44 : ... <http://vviale.developpez.com/tutoriels/definir-mot-passe/>

**Page 40**

- lien 45 : ... <http://fr.wikipedia.org/wiki/GNU>
- lien 46 : ... <http://fr.wikipedia.org/wiki/GNU/Linux>
- lien 47 : ... [http://fr.wikipedia.org/wiki/Mouvement\\_Emmaüs](http://fr.wikipedia.org/wiki/Mouvement_Emmaüs)
- lien 48 : ... <http://fr.wikipedia.org/wiki/GNU>
- lien 49 : ... <http://fr.wikipedia.org/wiki/GNU/Linux>
- lien 50 : ... <http://info.emmabuntus.org/>
- lien 51 : ... <http://www.emmabuntus.org/>
- lien 52 : ... <http://forum.emmabuntus.org/>
- lien 53 : ... <http://fr.wikipedia.org/wiki/Emmabuntüs>
- lien 54 : ... <http://emmabuntus.sourceforge.net/blog/category/reviews/>
- lien 55 : ... <http://fr.newspapers.emmabuntus.org/>

**Page 41**

- lien 56 : ... <http://www.netgear.fr/home/products/wireless-adapters/simplesharing/WNA1100.aspx>
- lien 57 : ... <http://www.netgear.fr/home/products/wireless-adapters/simplesharing/WNA1000M.aspx>
- lien 58 : ... <http://doc.ubuntu-fr.org/wifi>
- lien 59 : ... <http://doc.ubuntu-fr.org/hplip>
- lien 60 : ... <http://doc.ubuntu-fr.org/hplip>
- lien 61 : ... <http://doc.ubuntu-fr.org/imprimante>
- lien 62 : ... <http://doc.ubuntu-fr.org/simple-scan>
- lien 63 : ... <http://doc.ubuntu-fr.org/scanner>
- lien 64 : ... [http://doc.ubuntu-fr.org/cles\\_3g](http://doc.ubuntu-fr.org/cles_3g)
- lien 65 : ... <http://doc.ubuntu-fr.org/sfr>
- lien 66 : ... [http://doc.ubuntu-fr.org/orange\\_3g](http://doc.ubuntu-fr.org/orange_3g)
- lien 67 : ... [http://fr.wikipedia.org/wiki/Clementine\\_\(logiciel\)](http://fr.wikipedia.org/wiki/Clementine_(logiciel))
- lien 68 : ... <http://fr.wikipedia.org/wiki/Rhythmbox>

**Page 42**

- lien 69 : ... [http://fr.wikipedia.org/wiki/Cheese\\_\(logiciel\)](http://fr.wikipedia.org/wiki/Cheese_(logiciel))
- lien 70 : ... <http://doc.ubuntu-fr.org/webcam>
- lien 71 : ... <http://fr.wikipedia.org/wiki/Cairo-Dock>

**Page 43**

- lien 72 : ... <http://fr.wikipedia.org/wiki/Firefox>
- lien 73 : ... [http://fr.wikipedia.org/wiki/Chromium\\_\(navigateur\\_web\)](http://fr.wikipedia.org/wiki/Chromium_(navigateur_web))
- lien 74 : ... [http://fr.wikipedia.org/wiki/Adblock\\_Plus](http://fr.wikipedia.org/wiki/Adblock_Plus)

lien 75 : ... <http://fr.wikipedia.org/wiki/Ghostery>

#### Page 44

lien 76 : ... <http://www.commentcamarche.net/faq/35490-emmabuntus-2-installation-et-contrôle-parental>

lien 77 : ... <http://fr.wikipedia.org/wiki/LibreOffice>

lien 78 : ... <http://fr.wikipedia.org/wiki/AbiWord>

lien 79 : ... <http://fr.wikipedia.org/wiki/Gnumeric>

lien 80 : ... <http://fr.wikipedia.org/wiki/00o4Kids>

lien 81 : ... <http://fr.wikipedia.org/wiki/Arial>

lien 82 : ... <http://fr.wikipedia.org/wiki/Calibri>

lien 83 : ... <http://fr.wikipedia.org/wiki/Writer>

#### Page 45

lien 84 : ... <http://fr.wikipedia.org/wiki/ODT>

lien 85 : ... [http://fr.wikipedia.org/wiki/Microsoft\\_Office](http://fr.wikipedia.org/wiki/Microsoft_Office)

lien 86 : ... [http://fr.wikipedia.org/wiki/Microsoft\\_Word](http://fr.wikipedia.org/wiki/Microsoft_Word)

lien 87 : ... <http://fr.wikipedia.org/wiki/Audacious>

lien 88 : ... <http://fr.wikipedia.org/wiki/Kaffeine>

lien 89 : ... [http://fr.wikipedia.org/wiki/VLC\\_media\\_player](http://fr.wikipedia.org/wiki/VLC_media_player)

lien 90 : ... <http://fr.wikipedia.org/wiki/Youtube>

lien 91 : ... [http://fr.wikipedia.org/wiki/Adobe\\_Flash](http://fr.wikipedia.org/wiki/Adobe_Flash)

lien 92 : ... [http://fr.wikipedia.org/wiki/Adobe\\_Flash#Controverses](http://fr.wikipedia.org/wiki/Adobe_Flash#Controverses)

lien 93 : ... <http://fr.wikipedia.org/wiki/Skype>

lien 94 : ... [http://fr.wikipedia.org/wiki/Skype#S.C3.A9curit.C3.A9\\_et\\_pol.C3.A9miques](http://fr.wikipedia.org/wiki/Skype#S.C3.A9curit.C3.A9_et_pol.C3.A9miques)

lien 95 : ... <http://fr.wikipedia.org/wiki/Empathy>

lien 96 : ... <http://fr.wikipedia.org/wiki/Ekiga>

lien 97 : ... [http://fr.wikipedia.org/wiki/Paquet\\_\(logiciel\)](http://fr.wikipedia.org/wiki/Paquet_(logiciel))

lien 98 : ... <http://doc.ubuntu-fr.org/software-center>

lien 99 : ... <http://doc.ubuntu-fr.org/synaptic>

lien 100 : ... <http://doc.ubuntu-fr.org/apt-get>

#### Page 46

lien 101 : ... <http://doc.ubuntu-fr.org/>

lien 102 : ... <http://fr.wikipedia.org/wiki/Wine>

lien 103 : ... <http://fr.wikipedia.org/wiki/Notepad++>

lien 104 : ... <http://appdb.winehq.org/objectManager.php?sClass=category&iId=&sAction=view&sTitle=Browse+Applications>

lien 105 : ... <http://fr.wikipedia.org/wiki/PlayOnLinux>

lien 106 : ... <http://fr.wikipedia.org/wiki/VirtualBox>

lien 107 : ... [http://fr.wikipedia.org/wiki/Correspondance\\_entre\\_logiciels\\_libres\\_et\\_logiciels\\_propriétaires](http://fr.wikipedia.org/wiki/Correspondance_entre_logiciels_libres_et_logiciels_propriétaires)

#### Page 47

lien 108 : ... <http://fr.wikipedia.org/wiki/TeamViewer>

lien 109 : ... <http://fr.faq.emmabuntus.org/>

lien 110 : ... <http://forum.emmabuntus.org/>

lien 111 : ... <http://ideefixe.developpez.com/tutoriels/linux/premiers-pas-emmabuntus/>

#### Page 49

lien 112 : ... <http://alm.developpez.com/tutoriels/methodes-agiles/guide-lean-management-usage-equipes-agiles/satisfaire-client-comprendre-son-attente/>

lien 113 : ... <http://alm.developpez.com/tutoriels/methodes-agiles/guide-lean-management-usage-equipes-agiles/management-visuel-visualiser-challenge-problemes/>

lien 114 : ... <http://alm.developpez.com/tutoriels/methodes-agiles/guide-lean-management-usage-equipes-agiles/amelioration-continue-trouver-leviers-amelioration/>

lien 115 : ... <http://alm.developpez.com/tutoriels/methodes-agiles/guide-lean-management-usage-equipes-agiles/>