

# Developpez

## Magazine

Edition de Avril-Mai 2008.

Numéro 15.

Magazine en ligne gratuit.

Diffusion de copies conformes à l'original autorisée.

Réalisation : Baptiste Wicht et Alexandre Pottiez

Rédaction : la rédaction de Developpez

Contact : magazine@redaction-developpez.com

### Index

<a href="#">Java</a>	Page 2
<a href="#">PHP</a>	Page 11
<a href="#">(X)HTML/CSS</a>	Page 17
<a href="#">Flex</a>	Page 24
<a href="#">DotNet</a>	Page 29
<a href="#">C/C++/GTK/Qt</a>	Page 35
<a href="#">Linux</a>	Page 41
<a href="#">Mac</a>	Page 45
<a href="#">Conception</a>	Page 47
<a href="#">Liens</a>	Page 53

### Editorial

Après avoir profité de deux long week-end, profitez maintenant de ce nouveau magazine où vous découvrirez une série d'articles, de critiques de livres, de questions/réponses sur diverses technologies.

La rédaction

### Article Flex

## Des applications localisées sous Flex 2

*Réaliser des applications tenant compte de la langue de l'environnement d'exécution sous Flex2.*

par **Olivier Bugalotto**  
Page 24



### Article DotNet

## Introduction aux contrôles templates pour Asp.Net 2.0 en C#

*Cet article constitue une introduction à la création de contrôles templates pour Asp.Net en C#.*

par **Nico-pyright**  
Page 29





## Les derniers tutoriels et articles

### Création d'une application de type CRUD avec JSF et JPA

Cet article a pour objectif de vous présenter la tâche de création d'une application complète de type *CRUD* (*Create, Read, Update, Delete*) permettant d'effectuer les opérations de création, consultation, suppression et modification d'une entité et ce en utilisant le *framework* web *JSF* et le *framework* de persistance *JPA*.

#### 1. Introduction

Une application *CRUD* permet d'effectuer les opérations de listing, ajout, modification et suppression sur une entité donnée. Ce cas d'utilisation est si fréquent dans le développement logiciel qu'il est rare de trouver une application qui ne fasse pas du *CRUD*.

La mise en place d'une telle application nécessite de pouvoir effectuer les opérations *CRUD* sur une source de données (Base de données, fichier plat, etc.) et de fournir une interface graphique (client lourd ou web, etc.) pour réaliser ces opérations.

Le but de cet article est donc de présenter et expliquer la création d'une application web Java permettant de faire du *CRUD* sur une seule entité en utilisant *JSF* ([Lien1](#)) comme *framework* de présentation et *JPA* comme *framework* de persistance.

#### 2. Squelette de l'application

Il s'agit d'une application web Java. Si vous utilisez un EDI ([Lien2](#)) tel qu'*Eclipse* ou *NetBeans*, il est possible de générer automatiquement la structure d'une application web *JSF* grâce aux assistants de ces *EDIs*.

*Eclipse* ([Lien3](#)) prend en charge le développement d'applications *JSF* via le pack *Web Tools Project 2*.

Les étapes de configuration d'*Eclipse* et de la création d'un projet *JSF* sont présentées sous forme de démo *Flash* dans cet article ([Lien4](#)).

#### 2.1. Configuration de JSF

Ceci revient à :

- Ajouter une implémentation *JSF* au classpath ([Lien5](#)). Dans l'exemple fourni avec cet article, j'utilise la *Sun Reference Implementation 1.2.6 téléchargeable ici* ([Lien6](#)). Cette version supporte *JSF* 1.2.
- Déclarer la *Faces Servlet* dans *web.xml* et l'associer à *\*.jsf*. Cette servlet ([Lien7](#)) joue le rôle de la *Front Servlet* du MVC/Model 2 ([Lien8](#)).
- Ajouter le fichier *faces-config.xml* qui permet de configurer l'application *JSF* (déclaration des *managed-beans/controllers*, déclaration des règles de navigation, etc.)

Après avoir déclaré la *Faces Servlet* ainsi que son affectation aux urls de type *\*.jsf*, voici ce que devient *web.xml* : dans le reste de l'article :

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
        xmlns="http://java.sun.com/xml/ns/javaee"
        xmlns:web="http://java.sun.com/xml/ns/javaee/web
-app_2_5.xsd"
```

```
        xsi:schemaLocation="http://java.sun.com/xml/ns
/javaee http://java.sun.com/xml/ns/javaee/web-
app_2_5.xsd"
        id="WebApp_ID" version="2.5">
    <display-name>jsf-crud</display-name>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
    <servlet>
        <servlet-name>Faces Servlet</servlet-
name>
        <servlet-
class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-
name>
        <url-pattern>*.jsf</url-pattern>
    </servlet-mapping>
</web-app>
```

Et voici le fichier *faces-config.xml* (vide pour l'instant, mais il sera mis à jour au fur et à mesure de l'ajout de nouvelles fonctionnalités à l'application) :

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config
xmlns="http://java.sun.com/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns
/javaee
        http://java.sun.com/xml/ns/javaee/web-
facesconfig_1_2.xsd"
        version="1.2">
</faces-config>
```

#### 2.2. Configuration de JPA

La configuration de *JPA* consiste à :

- Ajouter une implémentation *JPA* au classpath. Dans le cadre de cet article, j'utilise *Toplink 2.41* comme implémentation qui est téléchargeable ici ([Lien9](#)).
- Créer le descripteur de persistance qui sert à spécifier les paramètres de connexion à la base de données (url de connexion, login, mot de passe, etc.) ainsi qu'à la déclaration des classes persistantes.

J'utilise *Toplink* comme implémentation *JPA* et *HSQldb* comme base de données. Il faut donc mettre *toplink.jar* et *hsqldb.jar* dans le *classpath* de l'application.

Il faut ensuite créer un descripteur de persistance **JPA** (*persistence.xml*) dans un dossier *META-INF* à la racine du dossier source.

```
<persistence
xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/p
ersistence
http://java.sun.com/xml/ns/persistence/persistence_1
_0.xsd"
version="1.0">

  <persistence-unit name="jsf-crud">
    <properties>
      <property
name="toplink.logging.level" value="INFO" />
      <property name="toplink.target-
database" value="HSQL" />
      <property
name="toplink.jdbc.driver"
value="org.hsqldb.jdbcDr
iver" />
      <property
name="toplink.jdbc.url"
value="jdbc:hsqldb:file:
test" />
      <property
name="toplink.jdbc.user" value="sa" />
      <property name="toplink.ddl-
generation"
value="create-tables" />
    </properties>
  </persistence-unit>
</persistence>
```

Pour l'instant, ce fichier ne contient que le nom de l'unité de persistance (*jsf-crud*) ainsi que les paramètres de connexion à la base de données qui sont:

- Pilote JDBC ([Lien10](#)): "org.hsqldb.jdbcDriver"
- URL de connexion: "jdbc:hsqldb:file:test" qui indique à **HSQLDB** de stocker la base de données dans un fichier nommé "test". Il est aussi possible de stocker la base dans la mémoire vive (*RAM*) en utilisant un chemin du type "jdbc:hsqldb:mem:test". Mais les données seraient alors perdues à l'arrêt de l'application.
- Login: "sa" (l'équivalent du root dans *HSQLDB*)
- Pilote JDBC: "org.hsqldb.jdbcDriver"
- *toplink.ddl-generation*: "create-tables" indique à *Toplink* de créer les tables si elles n'existent pas.

## 2.3.Couche métier

L'application qu'on désire développer permet de faire les opérations de création, modification, suppression et listing sur une seule entité. Normalement, la partie Model de l'application devrait être séparée en deux sous parties: **DAO** (*Data Access Object*) pour l'accès aux données et Service pour faire la logique métier. Mais dans le cadre de cet article, on se limitera à la couche *DAO* sans passer par la couche service vu que l'on n'a pas de logique métier à proprement parler.

### 2.3.1.Entité

L'entité sur laquelle on va travailler représente une personne. On se contentera de deux attributs, nom et prénom. Voici son code:

```
package model.dto;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

/**
 * Cette classe représente une personne. C'est une
 * entité persistente vu qu'on
 * l'a annoté avec l'annotation Entity.
 *
 * @author <a
href="mailto:djo.mos.contact@gmail.com">djo.mos</a>
 *
 */
@Entity
public class Person {
    private Long id;
    private String firstName;
    private String lastName;

    /**
     * C'est l'accessor de l'identifiant.<br />
     * On indique qu'un champ est un identifiant
     en l'annotant avec Id. <br />
     * De plus, si on ajoute GeneratedValue, alors
     c'est la base de données ou
     * l'implémentation JPA qui se charge
     d'affecter un identifiant unique.
     *
     * @return l'identifiant.
     */
    @Id
    @GeneratedValue
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
}
```

Comme vous le remarquez, cette classe est annotée avec **JPA** pour pouvoir être persistée ou retrouvée dans la base de données:

- L'annotation **@Entity** sur la classe **Person** pour la déclarer comme classe persistante. Cette annotation est obligatoire pour une classe persistante.
- L'annotation **@Id** sur le getter du champ id pour le déclarer comme l'identifiant. Cette annotation est obligatoire pour une classe persistante.
- L'annotation **@GeneratedValue** sur le getter du champ id pour déclarer que sa valeur est auto générée.

Notez que les champs *firstName* et *lastName* ne sont pas annotés. Ils seront pourtant persistés car dans un souci de simplification, **JPA** considère que tout champ d'une classe persistante est implicitement persistant, à moins qu'il soit annoté avec **@Transient**.

Il faut ensuite déclarer cette classe dans le descripteur de persistance *persistence.xml* pour qu'elle soit prise en charge:

```
<class>model.dto.Person</class>
```

### 2.3.2.DAO

On va maintenant encapsuler les opérations de persistance (création, modification, suppression et lecture) sur l'entité **Person** dans un **DAO**.

```
package model.dao;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.Persistence;

import model.dto.Person;

/**
 * C'est le DAO qui permet d'effectuer des opérations
 * portant sur une personne
 * dans la base de données.
 *
 * @author <a
 * href="mailto:djo.mos.contact@gmail.com">djo.mos</a>
 *
 */
public class PersonDao {
    private static final String JPA_UNIT_NAME =
"jsf-crud";
    private EntityManager entityManager;

    protected EntityManager getEntityManager() {
        if (entityManager == null) {
            entityManager =
Persistence.createEntityManagerFactory(
JPA_UNIT_NAME).c
reateEntityManager();
        }
        return entityManager;
    }
}
```

Le **DAO** qu'on va développer va utiliser l'**API** de **JPA** pour réaliser les opérations de persistance.

Pour pouvoir l'utiliser, il faut d'abord créer une instance de la classe **EntityManager**. Pour le faire, on passe par une fabrique (**Factory**) qu'on récupère via la méthode statique `Persistence.createEntityManagerFactory()` (Je sais, encore une autre fabrique :) ) en lui passant comme paramètre le nom de l'unité de persistance (le même déclaré dans *persistence.xml*):

```
<persistence-unit name="jsf-crud">
```

On verra plus tard comment utiliser l'**EntityManager** créée pour effectuer les opérations de persistance sur une entité.

Dans un souci de simplicité, la gestion de l'**EntityManager** dans cet article est faite à la main.

Mais dans la pratique, cette approche a de nombreuses limites, et on lègue la gestion de l'**EntityManager** à un conteneur tel que le serveur d'application ou encore un conteneur léger comme **Spring**. Il suffit alors d'indiquer au conteneur d'injecter l'**EntityManager**

créée dans nos **DAOs**.

### 2.4.Couche contrôle

Il s'agit ici de créer un **managed-bean JSF (Controller)** qui fera le lien entre les pages et la couche métier.

```
package control;

public class PersonCtrl {
}
```

Il faut ensuite déclarer cette classe dans *faces-config.xml* pour l'exposer aux pages **JSF**:

```
<managed-bean>
    <managed-bean-name>personCtrl</managed-bean-
name>
    <managed-bean-
class>control.PersonCtrl</managed-bean-class>
    <managed-bean-scope>session</managed-bean-
scope>
</managed-bean>
```

La déclaration d'un **managed-bean JSF** prend ces paramètres:

- **managed-bean-name**: Le nom sous lequel le bean sera accessible depuis les pages. En général, c'est le nom de la classe avec la première lettre en minuscule (C'est ce qui est fait dans ce cas: `personCtrl`). Mais rien ne vous empêche de le nommer comme bon vous semble.
- **managed-bean-class**: Le nom complet de la classe du **managed-bean**.
- **managed-bean-scope**: La durée de vie du **managed-bean (request, session, application, none)**. Dans ce cas, je l'ai mis à `session` car on aura des opérations sur plusieurs requêtes.

Une fois le **managed-bean** déclaré dans *faces-config.xml*, il devient alors accessible depuis les pages **JSF** via l'**EL (Expression Language)** "**#{nomDuBean}**".

## 3.Implémenter l'opération Read

### 3.1.Dans la couche DAO

Dans **PersonDao**, on ajoute une méthode `selectAll` qui retourne la liste de toutes les personnes depuis la base de données:

```
/**
 * L'opération Read
 * @return toutes les personnes dans la base
 * de données.
 */
public List<Person> selectAll() {
    List<Person> persons =
getEntityManager().createQuery(
"select p from Person
p").getResultList();
    return persons;
}
```

Notez que l'on accède à l'**EntityManager** via son `getter` pour s'assurer qu'il soit créé.

Comme cité plus haut, on utilise l'**EntityManager** pour exécuter une requête sur la base de données.

Notez que cette requête est exprimée en **JPA-QL (Java Persistence API Query Language)** et non pas en **SQL**. **JPA-SQL** est similaire à **SQL** mais est orienté Objet.

### 3.2. Dans la couche Control

On doit ensuite modifier le contrôleur pour offrir aux pages *JSF* la possibilité de lister les entités personnes. Dans la classe *PersonCtrl*, on ajoute une liste de personnes (pour stocker les personnes récupérées de la base de données) ainsi qu'une instance du *DAO* qu'on a créé (pour pouvoir exécuter les opérations de persistance sur la base de données):

```
private PersonDao pDao = new PersonDao();
private List<Person> persons;
```

Pour initialiser cette liste, mieux vaut éviter de le faire dans le constructeur du *managed-bean* vu que l'on n'est pas sûr de l'ordre d'initialisation des différents modules, et qu'il se peut que ce constructeur soit appelé alors que *JPA* ne soit pas encore initialisé. On va donc différer l'initialisation de cette liste dans son getter, de cette façon:

```
public List<Person> getPersons() {
    if(persons==null){
        persons = pDao.findAll();
    }
    return persons;
}
```

*JSF* accède **toujours** aux champs d'un *managed-bean* via les accesseurs et les mutateurs. On est donc sûr de passer par *getPersons* en la référençant depuis *JSF* via `"#{personCtrl.persons}"`.

### 3.3. Dans la couche View

On va maintenant afficher la liste des personnes dans une page *JSF* dans un format tabulaire. On crée une page *list.jsp* avec le contenu suivant:

```
<f:view>
    <h:dataTable border="0" rules="all"
value="##{personCtrl.persons}"
        var="p">
        <h:column>
            <f:facet name="header">
                <h:outputText
value="Prénom" />
            </f:facet>
            <h:outputText
value="#{p.firstName}" />
        </h:column>
        <h:column>
            <f:facet name="header">
                <h:outputText
value="Nom" />
            </f:facet>
            <h:outputText
value="#{p.lastName}" />
        </h:column>
    </h:dataTable>
</f:view>
```

On utilise le composant standard *dataTable* pour afficher la liste des personnes.

Ce composant prend entre autre les paramètres suivants:

- **value**: une *EL* désignant la liste à afficher. Dans ce cas, on désire afficher la liste des personnes définies dans le *managed-bean* *PersonCtrl*.
- **var**: *dataTable* va itérer sur la liste déclarée dans l'attribut *value*. A chaque itération, l'élément courant est mis dans une variable temporaire dont le nom est contrôlé par la valeur de l'attribut *var*.

- **border** et **rules** sont utilisées pour configurer l'affichage (pas de bordure, afficher des lignes autour de chaque cellule)

A l'inverse des autres taglibs (*Struts*, *JSTL*, etc.), le modèle de composants de *JSF* introduit une abstraction lors de l'itération sur une liste de données en définissant les colonnes au lieu des lignes.

Ici, on définit 2 colonnes:

- Une colonne pour afficher les prénoms des personnes: Ceci est fait via le composant **column**. Ce dernier répète son contenu à chaque itération sur la liste définie dans le **dataTable** parent.

Dans ce cas, le contenu de **column** qui est le composant **outputText** (qui permet d'afficher du texte) sera répété pour chaque personne de la liste des personnes dans une ligne de la table. **outputText** prend un attribut **value** qui indique le texte à afficher.

Ici, on a mis l'*EL* `"#{p.firstName}"` où *p* est la variable d'itération (déclarée dans l'attribut *var* du **dataTable** parent) qui est une instance de *Person*.

Le sous composant **facet** permet de définir une sous partie du composant parent (**column**) et ne sera pas répété pour chaque itération. Ici, le **facet** définit le contenu de l'entête de la colonne qui est le texte "Prénom".

- Une autre colonne pour afficher les noms des personnes. elle est définie de la même façon que la première colonne.

Il faut aussi créer une page *index.jsp* dont voici le contenu:

```
<body>
    <jsp:forward page="list.jsp" />
</body>
```

*index.jsp* est la page d'accueil vu qu'on l'a déclaré dans *web.xml*. Pour pouvoir accéder à une page *JSF*, il faut utiliser le suffixe *.jsf*.

C'est pour cette raison qu'on passe par l'élément *forward* qui permet de référence la page *JSF* avec le suffixe *.jsf*.

### 3.4. Test

En exécutant l'application, rien d'utile dans l'affichage vu qu'on n'a pas encore de données, il faut juste s'assurer qu'il n'y ait pas eu d'exceptions au démarrage.

Mais en suivant le *log* de l'application (dans la console d'*Eclipse* par exemple), on voit :

```
[TopLink Fine]: 2007.12.02 12:10:28.968
--ServerSession(3273383)
--Connection(15513215)--
Thread(Thread[http-8080-2,5,main])
--CREATE TABLE PERSON (ID NUMERIC(19) NOT NULL,
LASTNAME VARCHAR(255), FIRSTNAME VARCHAR(255), PRIMARY
KEY (ID))
:
:
[TopLink Fine]: 2007.12.02 12:10:29.318--
ServerSession(3273383)
--Connection(15513215)
--Thread(Thread[http-8080-2,5,main])
--SELECT ID, LASTNAME, FIRSTNAME FROM PERSON
```

Ce qui montre que *Toplink* a créé la table *PERSON* dans la base de données et qu'il a ensuite effectué un *SELECT* lors de l'affichage de la table.

## 4. Implémenter l'opération Create

### 4.1. Dans la couche DAO

Dans *PersonDao*, on ajoute la méthode suivante:

```
/**
 * L'opération Create
 * @param u La personne à insérer dans la base de
données.
 * @return La personne insérée
 */
public Person insert(Person u) {
    getEntityManager().getTransaction().begin();
    getEntityManager().persist(u);
    getEntityManager().getTransaction().commit();
    return u;
}
```

Cette méthode fait appel à l'API de *JPA* pour persister une personne dans la base de données.

Comme vous le remarquez, j'ai entouré l'opération *persist* par une ouverture d'une transaction et par son *commit*.

C'est une très mauvaise idée de s'y prendre de cette façon, i.e. gérer les transactions au niveau du *DAO*.

Dans une application réelle, on a en général plusieurs opérations qui doivent se réaliser d'une façon atomique.

Par exemple: on ajoute une entité dans la base de données et on ajoute un lien vers cette entité dans une autre table.

Ces deux opérations doivent se réaliser d'une façon atomique. C'est justement la raison d'être des transactions.

Or puisque les *DAO* offrent des fonctionnalités unitaires (*create*, *update*, etc.), il faut plutôt implémenter les transactions dans la couche *Service* et optimalement d'une façon déclarative en utilisant *Spring* par exemple.

Cette remarque vaut aussi pour les opérations *update* et *delete*.

### 4.2. Dans la couche Control

Dans *PersonCtrl*, on ajoute un champ de type *Person* qui servira à recueillir les informations de l'utilisateur.

```
private Person newPerson = new Person();
```

On ajoute aussi l'action *createPerson* dans le contrôleur *PersonCtrl* qui utilise le *DAO* et l'instance *newPerson* pour ajouter une personne dans la base de données.

Cette action doit être associée à un *submit* dans la page *JSF* d'ajout d'utilisateur.

```
public String createPerson() {
    pDao.insert(newPerson);
    newPerson = new Person();
    persons = pDao.selectAll();
    return "list";
}
```

Cette méthode ne fait qu'appeler la méthode *insert* du *DAO*.

Ensuite, elle crée à nouveau *newPerson* pour la prochaine insertion. Enfin, elle met à jour la liste des personnes pour refléter l'ajout de la nouvelle personne.

Pour retourner à la page de listing suite à la création d'une personne, l'action doit retourner un littéral ("*list*" par exemple dans ce cas). On doit ensuite ajouter une règle de navigation qui part de la page d'ajout d'une personne *add.jsp* et mène vers *list.jsp* suite à un résultat "*list*" dans *faces-config.xml*:

```
<navigation-rule>
    <display-name>add</display-name>
    <from-view-id>/add.jsp</from-view-id>
    <navigation-case>
        <from-outcome>list</from-outcome>
        <to-view-id>/list.jsp</to-view-id>
        <redirect />
    </navigation-case>
</navigation-rule>
```

Une règle de navigation *JSF* se configure ainsi:

- l'élément *from-view-id*: le nom de la page source.
- *navigation-case*: cas de navigation:
- *from-outcome*: le littéral qui active ce cas de navigation
- *to-view-id*: où aller si ce cas de navigation est activé.
- *redirect*: un élément optionnel qui s'il est présent, le cas de navigation effectue une *redirect* au lieu d'un *forward*.

Il est possible de déclarer plusieurs cas de navigation dans la même règle de navigation.

On peut voir ça comme ceci: les cas de navigation sont groupés par la page de départ dans une règle de navigation.

### 4.3. Dans la couche View

On crée une page *add.jsp*, qui permet de saisir les données d'une nouvelle personne et d'invoquer l'action *createPerson*:

```
<f:view>
    <h:form>
        <h:panelGrid border="0" columns="3"
cellpadding="5">
            <h:outputText value="Prénom" />
        >
            <h:inputText id="firstName"
value="#{personCtrl.newPerson.firstName}"
                required="true"
requiredMessage="Prénom obligatoire" />
            <h:message for="firstName" />

            <h:outputText value="Nom" />
            <h:inputText id="lastName"
value="#{personCtrl.newPerson.lastName}"
                required="true"
requiredMessage="Nom obligatoire" />
            <h:message for="firstName" />
            <h:outputText />
            <h:commandButton
value="Ajouter" action="#{personCtrl.createPerson}" />
        </h:panelGrid>
    </h:form>
</f:view>
```

Quelques notes:

- J'ai utilisé le composant *panelGrid* qui permet d'aligner ses fils en une grille pour la mise en page. Ici, j'ai défini 3 colonnes.
- J'ai utilisé le composant *inputText* qui permet de récupérer une valeur textuelle. L'attribut *value* de ce composant permet d'associer sa valeur avec un champ d'un *managed-bean* via une *EL*. Ici, je lie le premier champ textuel avec le champ *firstName* de *personCtrl.newPerson* et le second avec le champ *lastName*.
- J'ai ajouté des contraintes *required=true* pour les deux champs texte (nom et prénom) pour indiquer à *JSF* que leurs valeurs ne peuvent pas être vides ainsi que des éléments *message* pour afficher les messages d'erreur en cas d'erreur de validation.
- Enfin, un *commandButton* pour invoquer l'action *PersonCtrl.createPerson* (spécifié via une *EL* dans

l'attribut *action*).

Pour simplifier la navigation entre les deux pages *add.jsp* et *list.jsp*, on va ajouter un menu dans les deux pages dont voici le code:

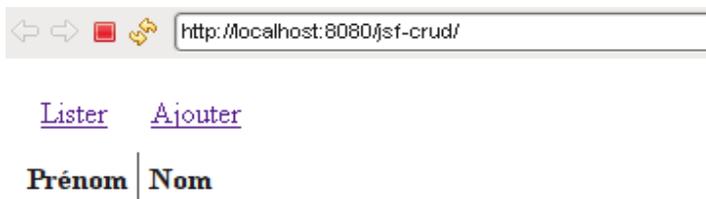
```
<h:panelGrid columns="2" cellpadding="10">
  <h:outputLink value="list.jsf">
    <h:outputText value="Lister" />
  </h:outputLink>

  <h:outputLink value="add.jsf">
    <h:outputText value="Ajouter" />
  </h:outputLink>
</h:panelGrid>
```

Ce ne sont que deux liens hypertexte dans une grille de 2 colonnes. Ce menu est à mettre dans les deux pages *add.jsp* et *list.jsp* au tout début de la page juste après le composant *<f:view>*.

#### 4.4.Test

On effectue un test en ajoutant quelques personnes: Initialement la liste est vide:



En cliquant sur le lien "Ajouter", on passe à la page d'ajout:



Après avoir rempli et validé le formulaire, on revient à la liste des personnes:



## 5. Implémenter l'opération Delete

### 5.1. Dans la couche DAO

On commence par ajouter une méthode delete à *PersonDao*:

```
/**
 * L'opération Delete
 * @param u La personne à supprimer de la base de
 * données.
 */
public void delete(Person u) {
    getEntityManager().getTransaction().begin();
    u = getEntityManager().merge(u); //<-Important
    getEntityManager().remove(u);
    getEntityManager().getTransaction().commit();
}
```

La remarque mentionnée ici ([Lien11](#)) s'applique aussi ici.

L'instruction *"u = em.merge(u)"* est très importante, sans elle, la méthode *deletePerson* risque de déclencher une exception car l'instance qu'on lui aura passé est détachée de la session de persistance.

La méthode *merge* de la classe *EntityManager* s'occupe de rattacher une entité à la session de persistance en cours.

Cette étape est inutile dans un environnement managé (où l'*EntityManager* est géré par le conteneur plutôt que par le développeur).

### 5.2. Dans la couche Control

Pour implémenter la fonction supprimer, on ajoute généralement un lien ou un bouton supprimer dans chaque ligne de la liste d'affichage. Suite à un clic sur le bouton supprimer, on invoque d'action delete qui doit récupérer la ligne sélectionnée et la supprimer.

Pour implémenter facilement un fonctionnement pareil dans *JSF*, on utilise un *DataModel* comme conteneur de la liste des valeurs et non plus une liste simple (*java.util.List*). En effet, dans le code d'une action donnée, *DataModel* permet de récupérer à tout moment la ligne ayant déclenché l'action. On remplace donc dans *PersonCtrl* le champ *persons* de type *List* et son accesseur par:

```
private DataModel persons;

public DataModel getPersons() {
    if (persons == null) {
        persons = new ListDataModel();
        persons.setWrappedData(pDao.selectAll());
    }
    return persons;
}
```

*DataModel* est une interface tandis que *ListDataModel* est une implémentation (tout comme pour *List* et *ArrayList*). Pour spécifier les éléments du *DataModel*, on passe une liste ordinaire (*java.util.List*, *java.util.Set*) comme paramètre à la méthode *setWrappedData*.

Il faut aussi mettre à jour la méthode *createPerson* pour refléter l'utilisation de *DataModel* au lieu de *List*:

```
public String createPerson() {
    pDao.insert(newPerson);
    newPerson = new Person();
    persons.setWrappedData(pDao.selectAll());
    return "list";
}
```

Voici ensuite la méthode *deletePerson* du contrôleur:

```
/**
 * L'opération de suppression à invoquer suite à la
 * sélection d'une
 * personne.
 *
 * @return outcome pour la navigation entre les pages.
 * Dans ce cas, c'est null pour
 * rester dans la page de listing.
 */
public String deletePerson() {
    Person p = (Person) persons.getRowData();
    pDao.delete(p);
    persons.setWrappedData(pDao.selectAll());
    return null;
}
```

La première ligne récupère la personne correspondant à la ligne sélectionnée dans la table. Reste plus qu'à invoquer la méthode *delete* du *DAO* et à mettre à jour la liste des personnes pour refléter la suppression.

### 5.3. Dans la couche View

Coté présentation, on ajoute une nouvelle colonne à la table des personnes dans *list.jsp* qui contient un bouton delete sur chaque ligne:

```
<h:column>
    <f:facet name="header">
        <h:outputText value="Opérations" />
    </f:facet>
    <h:commandButton value="Supprimer"
        action="#{personCtrl.deletePerson}" />
</h:column>
```

Il ne faut pas oublier d'englober le *dataTable* dans un *<h:form>* vu que l'on a des *commandButton*.

## 5.4. Test



En cliquant sur supprimer dans une ligne de la table, la personne correspondante est supprimée de la base de données et la liste est mise à jour

## 6. Implémenter l'opération Update

### 6.1. Dans la couche DAO

Dans *PersonDao*, on ajoute la méthode *update* que voici:

```
/**
 * L'opération Update
 * @param u La personne à mettre à jour dans la base
 * de données.
 * @return La personne mise à jour
 */
public Person update(Person u) {
    getEntityManager().getTransaction().begin();
    u = getEntityManager().merge(u);
    getEntityManager().getTransaction().commit();
    return u;
}
```

Les remarques mentionnées ici ([Lien11](#)) et ici ([Lien12](#)) s'appliquent aussi ici.

Normalement, une mise à jour est réalisée de la même façon qu'une insertion, c'est à dire avec la méthode *persist* de la classe *EntityManager*. *JPA* s'occupe ensuite de décider s'il s'agit d'une nouvelle entité et qu'il faut l'ajouter ou d'une entité déjà ajoutée et qu'il faut plutôt la mettre à jour.

Mais pour les mêmes raisons que pour la méthode *delete*, c'est à dire pour éviter le cas où l'entité passée est détachée, on utilise la méthode *merge* de la classe *EntityManager* qui rattache une entité à la session de persistance en cours tout en intégrant les modifications qu'elle a pu subir.

### 6.2. Dans la couche Control

Dans le contrôleur *PersonCtrl*, on ajoute un champ *editPerson* de type *Person* qui servira à recueillir les données saisies par l'utilisateur ainsi que son accesseur:

```
private Person editPerson;
```

Vous remarquerez qu'à l'inverse de *newPerson*, on n'a pas initialisé *editPerson* et qu'on l'a laissé à null.

En effet, *editPerson* sera utilisée dans la page *edit.jsp*, mais avant, on lui affectera la valeur de la personne sélectionnée pour édition dans la table des personnes.

Dans le contrôleur, on ajoute aussi la méthode `editPerson` qui sera appelée quand on clique sur le bouton modifier d'une ligne. Cette méthode affecte à `editPerson` la personne sélectionné et redirige l'utilisateur vers la page `edit.jsp`:

```
/**
 * Opération intermédiaire pour récupérer la personne à
 * modifier.
 * @return outcome pour la navigation entre les pages.
 * Dans ce cas, c'est "edit" pour
 * aller à la page de modification.
 */
public String editPerson() {
    editPerson = (Person) persons.getRowData();
    return "edit";
}
```

ainsi que la méthode qui effectue la mise à jour:

```
/**
 * L'opération Update pour faire la mise à jour.
 * @return outcome pour la navigation entre les pages.
 * Dans ce cas, c'est "list" pour
 * retourner à la page de listing.
 */
public String updatePerson() {
    pDao.update(editPerson);
    persons.setWrappedData(pDao.selectAll());
    return "list";
}
```

On ajoute alors la règle de navigation suivante:

```
<navigation-rule>
  <from-view-id>/list.jsp</from-view-id>
  <navigation-case>
    <from-outcome>edit</from-outcome>
    <to-view-id>/edit.jsp</to-view-id>
    <redirect />
  </navigation-case>
</navigation-rule>
```

Cette règle permet à partir de la page `list.jsp` d'aller à la page `edit.jsp` si on a un résultat `"edit"`

On ajoute aussi la règle de navigation suivante:

```
<navigation-rule>
  <from-view-id>/edit.jsp</from-view-id>
  <navigation-case>
    <from-outcome>update</from-outcome>
    <to-view-id>/list.jsp</to-view-id>
    <redirect />
  </navigation-case>
</navigation-rule>
```

Cette règle permet à partir de la page `edit.jsp` d'aller à la page `list.jsp` si on a un résultat `"update"`

### 6.3. Dans la couche View

On ajoute aussi le bouton modifier dans la page `list.jsp`. Ce bouton sera ajouté dans la même colonne (opérations) que le bouton supprimer et il invoque la méthode `PersonCtrl.editPerson`:

```
<h:column>
  <f:facet name="header">
    <h:outputText value="Opérations" />
  </f:facet>
```

```
<h:commandButton value="Modifier"
  action="#{personCtrl.editPerson}" />
<h:commandButton value="Supprimer"
  action="#{personCtrl.deletePerson}" />
</h:column>
```

On crée la page `edit.jsp` qui est exactement similaire à `add.jsp` avec la seule différence qu'elle pointe vers `editPerson` au lieu de `newPerson` et qu'elle invoque `updatePerson` au lieu de `createPerson`:

```
<f:view>
  <h:panelGrid columns="2" cellpadding="10">
    <h:outputLink value="list.jsp">
      <h:outputText value="Lister" />
    </h:outputLink>

    <h:outputLink value="add.jsp">
      <h:outputText
value="Ajouter" />
    </h:outputLink>
  </h:panelGrid>
  <h:form>
    <h:panelGrid border="0" columns="3"
cellpadding="5">
      <h:outputText value="Prénom" />
      <h:inputText id="firstName"
value="#{personCtrl.ed
itPerson.firstName}" required="true"
requiredMessage="Préno
m obligatoire" />
      <h:message for="firstName" />
      <h:outputText value="Nom" />
      <h:inputText id="lastName"
value="#{personCtrl.editPerson.lastName}"
required="true"
requiredMessage="Nom obligatoire" />
      <h:message for="firstName" />
      <h:outputText />
      <h:commandButton value="Mettre
à jour"
action="#{personCtrl.u
pdatePerson}" />
    </h:panelGrid>
  </h:form>
</f:view>
```

### 6.4. Test

On commence par la page `list.jsp`:

Prénom	Nom	Opérations
djo	mos	Modifier Supprimer
test	test	Modifier Supprimer

Comme vous le remarquez, on voit un bouton modifier dans

chaque ligne de la table.



En validant, la personne est mise à jour dans la base de données et on revient à la page *list.jsp* où on voit la personne mise à jour:



Prénom	Nom	Opérations	
jawher	moussa	Modifier	Supprimer
test	test	Modifier	Supprimer

En cliquant sur un de ces boutons, on passe à la page *edit.jsp* qui permet d'éditer la personne sélectionnée: On entre de nouvelles valeurs pour le nom et le prénom de la personne, ce qui donne:



## 7. Conclusion

Dans cet article, j'ai essayé de montrer d'une façon plus ou moins détaillée de montrer comment créer une application web de type *CRUD* qui utilise *JSF* comme *framework* web (présentation et contrôle) et *JPA* comme *framework* de persistance tout en expliquant au maximum chacune des étapes.

Il faut cependant noter que dans un souci de simplicité, j'ai dû m'y prendre d'une façon non-optimale et non-conforme aux design patterns en vigueur dans ce genre d'applications.

Les points à soigner sont les suivants:

- Absence de la couche service
- Gestion manuelle de l'*EntityManager*
- Gestion manuelle des transactions

Je compte montrer comment régler ces points en utilisant *Spring* dans un autre article.

Retrouvez l'article de Jawher Moussa en ligne : [Lien13](#)

## Les derniers tutoriels et articles

### Spreadsheet\_Excel\_Writer en PHP

Une des demandes les plus fréquentes dans les développements PHP, c'est la génération de fichiers Excel. Dans cet article nous allons générer des fichiers Excel sans passer par la méthode des CSV, deCOM ([Lien14](#)), ni même de Microsoft Excel ou OpenOffice.org. Tout cela en passant par un fork d'une librairie Perl.

#### 1. Introduction

##### 1. Avant-Propos

Dans le monde Perl, il existe une librairie qui permet la génération de fichiers Excel, cette librairie se nomme Spreadsheet::WriteExcel ([Lien15](#)), elle fut créée par JohnMcNamara ([Lien16](#)). Cette librairie a été adaptée dans le monde PHP, 2 fois. En effet il existe 2 fois la même librairie écrite un peu différemment.

Vous trouverez les sources et fichiers exemples de cet article ici ([Lien17](#))

#### 2. Les librairies PHP

##### 2.1. php\_writeexcel de Johann Hanne

Cette librairie est donc un portage de la classe Perl, vous pouvez la télécharger ici ([Lien18](#)). La dernière mise à jour date d'avril 2002. Cette classe est déjà relativement complète, elle s'inspire directement de la classe Perl, jusqu'à la convention de nommage. Pour la documentation Johann Hanne renvoie à la doc officielle de Perl (en anglais uniquement). Malheureusement toutes les méthodes ne sont pas implémentées et il faut se contenter des exemples mis sur son site pour s'inspirer et générer un fichier Excel.

##### 2.2. Spreadsheet\_Excel\_Writer de Xavier Noguier

Cette librairie est également un portage de la classe Perl, vous pouvez la télécharger ici ([Lien19](#)). La dernière version bêta date de septembre 2006, celle-ci semble plus complète. Une documentation multilingue se trouve sur le site de PEAR ([Lien20](#)). La différence se joue surtout au niveau de la convention de nommage qui ne reprend pas de Perl, en fait elle supprime l'underscore (\_), dans le nom de la méthode. La documentation ne fait pas référence à toutes les méthodes, en effet dans le package Perl on trouve de nombreux exemples, que l'on peut s'amuser à faire en PHP (vous en trouverez d'ailleurs pas mal dans les sources), et en réalisant quelques portages on se rend compte que certaines méthodes fonctionnent, alors qu'elles ne sont pas dans la documentation (la méthode sheet() notamment).

##### 2.3. Besoin complémentaire

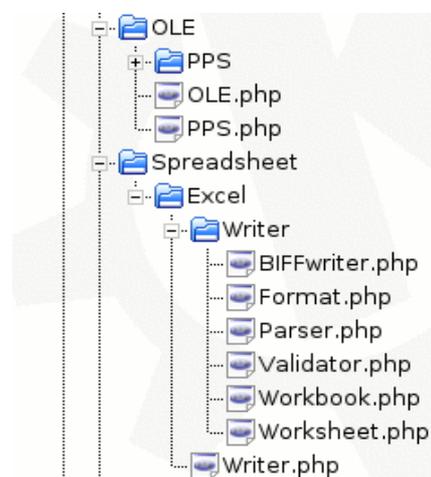
Pour réaliser nos fichiers Excel, nous aurons également besoin de la librairie OLE de PEAR. Tandis que Johann Hanne l'inclut directement dans sa librairie, pour utiliser la librairie de Xavier Noguier il faudra la télécharger sur le site de PEAR ([Lien21](#))

#### 3. Notre premier fichier

J'ai une préférence pour le travail de Xavier Noguier, et c'est donc à partir de ses classes que je vais développer ce petit article.

##### 3.1. La classe de base

###### 3.1.1. La structure



Organisation de la librairie

Un des avantages de la librairie de Xavier Noguier, est un projet PEAR, et par conséquent il utilise une convention de nommage strict, qui est également repris par le projet Zend Framework. Dès lors l'intégration de cette librairie dans un projet Zend Framework ne pose aucun problème.

Le nom de classe choisi par l'auteur de la classe ne respecte pas totalement les conventions établies par PEAR. La classe **Spreadsheet\_Excel\_Writer** devrait ainsi s'appeler par exemple **Spreadsheet\_Excel\_Writer\_Workbook\_Main**

#### Classe de base

```
class Spreadsheet_Excel_Writer extends Spreadsheet_Excel_Writer_Workbook
{
    /**
     * The constructor. It just creates a Workbook
     *
     * @param string $filename The optional filename for the Workbook, $filename create temporary file.
     * @return Spreadsheet_Excel_Writer_Workbook The Workbook created
     */
    function Spreadsheet_Excel_Writer($filename='tmp.xls')
    {
```

```

        $this->Spreadsheet_Excel_Writer_Workbook($filename);
    }

    /**
     * Send HTTP headers for the Excel file.
     *
     * @param string $filename The filename to use for
     HTTP headers
     * @access public
     */
    function send($filename)
    {
        header("Content-type:
application/vnd.ms-excel");
        header("Content-Disposition:
attachment; filename=\"$filename\"");
        header("Expires: 0");
        header("Cache-Control: must-revalidate,
post-check=0,pre-check=0");
        header("Pragma: public");
    }

    /**
     * Utility function for writing formulas
     * Converts a cell's coordinates to the A1 format.
     *
     * @access public
     * @static
     * @param integer $row Row for the cell to convert
(0-indexed).
     * @param integer $col Column for the cell to
convert (0-indexed).
     * @return string The cell identifier in A1 format
     */
    function rowcolToCell($row, $col)
    {
        if ($col > 255) { //maximum column
value exceeded
                return new PEAR_Error("Maximum
column value exceeded: $col");
        }

        $int = (int)($col / 26);
        $frac = $col % 26;
        $chr1 = '';

        if ($int > 0) {
            $chr1 = chr(ord('A') + $int -
1);
        }

        $chr2 = chr(ord('A') + $frac);
        $row++;

        return $chr1 . $chr2 . $row;
    }
}

```

Cette classe comme expliqué sur la documentation de PEAR est la ligne d'entrée, pour créer notre fichier Excel. Malheureusement elle ne m'a pas donné que des satisfactions, surtout intégrée dans un projet Zend\_Framework. J'ai donc quelque peu modifié cette classe en m'inspirant de la méthode de Johann Hanne, qui consiste à créer un fichier temporaire et à le lire.

#### Classe modifiée

```

class Spreadsheet_Excel_Writer extends
Spreadsheet_Excel_Writer_Workbook

```

```

{
    private $_filename;

    /**
     * The constructor. It just creates a Workbook
     *
     * @param string $filename The optional filename for
the Workbook, $filename create temporary file.
     * @return Spreadsheet_Excel_Writer_Workbook The
Workbook created
     */
    function Spreadsheet_Excel_Writer($filename =
'test.xls')
    {
        $this->Spreadsheet_Excel_Writer_Workbook($filename);
    }

    /**
     * Send HTTP headers for the Excel file.
     *
     * @param string $filename The filename to use for
HTTP headers
     * @access public
     */
    function send($filename='fichier.xls')
    {
        header("Content-type:
application/vnd.ms-excel");
        header("Content-Disposition:
attachment; filename=\"$filename\"");
        header("Expires: 0");
        header("Cache-Control: must-revalidate,
post-check=0,pre-check=0");
        header("Pragma: public");
    }

    /**
     * Utility function for writing formulas
     * Converts a cell's coordinates to the A1 format.
     *
     * @access public
     * @static
     * @param integer $row Row for the cell to convert
(0-indexed).
     * @param integer $col Column for the cell to
convert (0-indexed).
     * @return string The cell identifier in A1 format
     */
    function rowcolToCell($row, $col)
    {
        if ($col > 255) { //maximum column
value exceeded
                return new PEAR_Error("Maximum
column value exceeded: $col");
        }

        $int = (int)($col / 26);
        $frac = $col % 26;
        $chr1 = '';

        if ($int > 0) {
            $chr1 = chr(ord('A') + $int -
1);
        }

        $chr2 = chr(ord('A') + $frac);
        $row++;

        return $chr1 . $chr2 . $row;
    }

    /**
     * Read file create temporary file
     */
}

```

```

* @access public
*
*/
function sendFile()
{
    readfile($this->_filename);
    unlink($this->_filename);
}

```

Concrètement, qu'est ce que j'ai changé ?

J'ai ajouté une méthode pour la lecture et la destruction du fichier temporaire

```

function sendFile()
{
    readfile($this->_filename);
    unlink($this->_filename);
}

```

### 3.2. Notre premier fichier Excel

Exemple ici ([Lien22](#))

```

set_time_limit(300);

require_once 'Spreadsheet/Excel/Writer.php';

$workbook = new Spreadsheet_Excel_Writer();
$workbook->setTempDir('./tempdoc');
$workbook->send('base.xls');
$worksheet = $workbook->addWorksheet();

$worksheet->write(1,2,'toto');

$workbook->close();
$workbook->sendFile();

```

Examinons ce fichier ligne à ligne.

```

set_time_limit(300);

```

Set\_time\_limit(), permet de dépasser le temps d'attente qui généralement est limité à 30 secondes, ce qui peut-être nécessaire pour le fichier suivant ([Lien23](#)) par exemple, et qu'on augmente la boucle dans de plus grandes proportions. Malheureusement, pour des raisons de sécurité la fonction est désactivée chez Developpez.com, je vais donc la supprimer de mes exemples. Heureusement, très peu de scripts ont besoin de Set\_time\_limit().

```

require_once 'Spreadsheet/Excel/Writer.php';

```

On inclut le fichier de la classe Writer, pour pouvoir instancier un objet Spreadsheet\_Excel\_Writer().

Cette ligne est inutile dans un projet Zend Framework si vous utilisez Zend\_Loader::registerAutoload().

```

$workbook = new Spreadsheet_Excel_Writer();

```

Nous créons une instance de notre classe, nous pouvons si nous le voulons lui passer en paramètre le nom d'un fichier temporaire.

```

$workbook->setTempDir('./tempdoc');

```

Définit le dossier temporaire à utiliser pour le fichier OLE. Utilisez cette méthode si vous n'avez pas le droit d'écrire dans le dossier temporaire par défaut.

```

$workbook->send('base.xls');

```

Nous envoyons les entêtes qui permettent de dire aux navigateurs que nous voulons un fichier Excel, ici également vous pouvez passer en paramètre le nom que votre fichier portera.

Si nous arrêtons notre script à ce stade nous n'aurions qu'une application Excel, qui s'ouvrirait complètement vide.

```

$worksheet = $workbook->addWorksheet();

```

Ici, nous créons notre première Feuille(sheet), si vous désirez donner un nom à l'onglet de la feuille, vous pouvez lui passer ce nom en paramètre addWorksheet('Feuille1').

Pour plus de détails sur les Feuilles, reportez-vous au chapitre 3.

```

$worksheet->write(1,2,'toto');

```

Ici nous écrivons dans une cellule.

a méthode write prends trois paramètres minimum :

- La ligne du document Excel (cela commence à zéro)
- La colonne du document Excel (cela commence à zéro)
- Ce que l'on veut mettre dans la cellule (texte, formule etc..)

Dans le cas ici, nous afficherons donc dans la cellule 2C, le mot : 'toto' Voici ([Lien23](#)) de quoi vous faciliter la navigation dans les cellules.

Pour plus de détails sur les cellules, reportez-vous au chapitre 4.

```

$workbook->close();

```

Nous fermons l'objet

```

$workbook->sendFile();

```

Nous envoyons la lecture de notre fichier.

Nous venons de réaliser notre premier fichier Excel.

### 4. Feuilles (sheet)

Dans le fichier de base, nous appelons une méthode "addWorksheet()", afin d'avoir une feuille Excel (espace de travail).

Dans cette feuille nous pouvons passer certains paramètres de configuration via des méthodes setter.

Voici les paramètres :

- setPortrait : Définit l'orientation de la page en tant que portrait
- setLandscape : Définit l'orientation de la page en tant que paysage
- setPaper : Définit le type de papier. Ex. 1 = US Letter, 9 = A4
- setHeader : Définit l'en-tête de la page et, optionnellement, la marge
- setFooter : Définit le pied de page et, optionnellement, la marge
- centerHorizontally : Centre la page horizontalement
- centerVertically : Centre la page verticalement
- setMargins : Définit toutes les marges de la page à la même valeur, en pouces
- setMargins\_LR : Définit les marges gauches et droites à la même valeur, en pouces
- setMargins\_TB : Définit les marges supérieures et inférieures à la même valeur, en pouces
- setMarginLeft : Définit la marge gauche, en pouces
- setMarginRight : Définit la marge droite, en pouces

- setMarginTop : Définit la marge supérieure en pouces
- setMarginBottom : Définit la marge inférieure, en pouces

```
<?php

//set_time_limit(300);

require_once 'Spreadsheet/Excel/Writer.php';

$workbook = new Spreadsheet_Excel_Writer();
$workbook->send('worksheets.xls');
$worksheet = $workbook-
>addWorksheet('TestDeWorksheet');
$worksheet->setPaper(9);//Définit une page A4
$worksheet->setHeader('Mon beau fichier
Excel');//Définit un entête ,faites appercu pour voir
l'entête
$worksheet->setLandscape ();//Définit une
orientation Paysage.

$worksheet->write(0,0,utf8_decode('première
cellule d\'une page A4,paysage '));
$worksheet->write(31,10,utf8_decode('dernière
cellule d\'une page A4,paysage '));

$workbook->close();
$workbook->sendFile();

?>
```

Nous créons donc une feuille.

```
$worksheet = $workbook-
>addWorksheet('TestDeWorksheet');
```

Et nous passons par les méthodes setter, nos paramètres

```
$worksheet->setPaper(9);//Définit une page A4
$worksheet->setHeader('Mon beau fichier
Excel');//Définit un entête ,faites appercu pour voir
l'entête
$worksheet->setLandscape ();//Définit une orientation
Paysage.
```

Exemple ([Lien24](#))

## 5. Prenons soin de notre cellule

Notre librairie ne se limite pas à écrire dans des sheets ou des cellules, nous pouvons y mettre quelques options intéressantes

### 5.1. Array d'options

```
<?php
require_once "Spreadsheet/Excel/Writer.php";
$workbook = new Spreadsheet_Excel_Writer();
$workbook->send('format.xls');
$format = $workbook->addFormat(
    array(
        'Size' => 10,//taille du texte
        'Align' => 'center',//alignement du texte
        'Color' => 'white',//couleur du texte
        'FgColor' => 'magenta');//couleur du fond de
cellule
$worksheet = $workbook->addWorksheet();
$worksheet->write(1, 0, "Bonjour en Magenta !",
$format);
$workbook->close();
$workbook->sendFile();
?>
```

Cet exemple provient directement de la doc PEAR, cela va nous créer un fichier Excel avec une seule cellule écrite avec une taille 10 en blanc sous fond magenta. Concrètement, nous créons une nouvelle variable \$format dans laquelle nous appelons la méthode "addformat" et lui passons un tableau de paramètres.

Cette variable sera passer en 4ième argument de la méthode write.

Définition d'une cellule formatée :

```
$format = $workbook->addFormat(
    array(
        'Size' => 10,//taille du texte
        'Align' => 'center',//alignement du texte
        'Color' => 'white',//couleur du texte
        'FgColor' => 'magenta');//couleur du fond de
cellule
```

Nous passons la variable en argument :

```
$worksheet->write(1, 0, "Bonjour en Magenta !",
$format);
```

Voici les paramètres possibles.

- Align : Définit l'alignement de la cellule
- VAlign : Définit l'alignement de la cellule
- HAlign : Définit l'alignement de la cellule
- Merge : Cette méthode est un alias de la méthode non-intuitive setAlign('merge')
- Bold : Définit l'intensité de la mise en gras d'un texte
- Bottom : Définit l'épaisseur de la bordure inférieure de la cellule
- Top : Définit l'épaisseur de la bordure supérieure de la cellule
- Left : Définit l'épaisseur de la bordure gauche de la cellule
- Right : Définit l'épaisseur de la bordure droite de la cellule
- Border : Définit les bordures d'une cellule dans un même style
- BorderColor : Définit toutes les bordures de la cellule à une même couleur
- BottomColor : Définit la couleur de la bordure inférieure de la cellule
- TopColor : Définit la couleur de la bordure supérieure de la cellule
- LeftColor : Définit la couleur de la bordure gauche de la cellule
- RightColor : Définit la couleur de la bordure droite de la cellule
- FgColor : Définit la couleur de premier-plan de la cellule
- BgColor : Définit la couleur d'arrière-plan de la cellule
- Color : Définit la couleur pour le contenu de la cellule
- Pattern : Définit les attributs d'une cellule
- Underline : Définit le soulignement du texte
- Italic : Définit le style de la police de caractère en italique
- Size : Définit la taille de la police de caractères
- TextWrap : Définit le retour automatique à la ligne du texte
- TextRotation : Définit l'orientation du texte
- NumFormat : Définit le format numérique
- StrikeOut : Définit la police de caractères comme barrée
- OutLine : Définit le dessin d'une police de caractères
- Shadow : Définit la police de caractères comme ombrée
- Script : Définit le type de script pour le texte
- FontFamily : Définit la famille de police de caractères

Nous pouvons également combiner les formats en créant des arrays et en utilisant `array_merge()`;  
exemple ici ([Lien25](#)).

```
<?php
require_once "Spreadsheet/Excel/Writer.php";
$workbook = new Spreadsheet_Excel_Writer();
$workbook->send('format.xls');
$couleur = array(
    Color => 'white',//couleur du texte
    FgColor => 'magenta');//couleur du fond de
cellule
$taille = array(
    Size => 10,//taille du texte
    Align => 'center');//alignement du texte

$Merge = array_merge($couleur,$taille);

$format1 = $workbook->addformat($taille);
$format2 = $workbook->addformat($couleur);
$formatMerge = $workbook->addformat($Merge);
$worksheet = $workbook->addWorksheet();
$worksheet->write(1, 0, "format 1", $format1);
$worksheet->write(2, 0, "format 2", $format2);
$worksheet->write(3, 0, "format merge ", $formatMerge);
$workbook->close();
$workbook->sendFile();
?>
```

## 5.2. MaitrePylos à l'envers

Mais le plus simple dans le format de cellule est d'utiliser son setter, en effet chaque propriété peut être appelée séparément et ajoutée à 'addformat()'.

---

```
<?php
require_once "Spreadsheet/Excel/Writer.php";
$workbook = new Spreadsheet_Excel_Writer();
$workbook->send('format.xls');

$worksheet = $workbook->addWorksheet();

$format= $workbook->addformat();
$format->setTextRotation(90);
$worksheet->write(1, 0, "Maitre", $format);

$format1= $workbook->addformat();
$format1->setTextRotation(270);
$worksheet->write(2, 0, "Pylos", $format1);

$format2 = $workbook->addformat();
$format2->setTextRotation(-1);
$format2->setFgColor(11);

$worksheet->write(2, 2, "MaitrePylos", $format2);

$workbook->close();
$workbook->sendFile();
?>
```

exemple ([Lien26](#))

## 5.3. Les formules

Un fichier Excel sans formule mathématique ne serait pas un fichier Excel.

Il suffit de noter "=SUM(A1,B2)", par exemple pour avoir l'addition.

```
require_once 'Spreadsheet/Excel/Writer.php';
```

```
$workbook = new Spreadsheet_Excel_Writer();
$workbook->send('stats.xls');
$worksheet = $workbook->addWorksheet();
# Set the column width for columns 1
$worksheet->setColumn(0, 0, 20);

# Create a format for the headings
$format = $workbook->addFormat();
$format->setBold();
```

```
# Write the sample data
$worksheet->write(0, 0, 'Sample', $format);
$worksheet->write(1, 0, 1);
$worksheet->write(2, 0, 2);
$worksheet->write(3, 0, 3);
$worksheet->write(4, 0, 4);
$worksheet->write(5, 0, 5);
$worksheet->write(6, 0, 6);
$worksheet->write(7, 0, 7);
$worksheet->write(8, 0, 8);
```

```
$worksheet->write(0, 1, 'Length', $format);
$worksheet->write(1, 1, 25.4);
$worksheet->write(2, 1, 25.4);
$worksheet->write(3, 1, 24.8);
$worksheet->write(4, 1, 25.0);
$worksheet->write(5, 1, 25.3);
$worksheet->write(6, 1, 24.9);
$worksheet->write(7, 1, 25.2);
$worksheet->write(8, 1, 24.8);
```

```
# Write some statistical functions
$worksheet->write(0, 4, 'Count', $format);
$worksheet->write(1, 4, '=COUNT(B2:B9)');
```

```
$worksheet->write(0, 5, 'Sum', $format);
$worksheet->write(1, 5, '=SUM(B2:B9)');
```

```
$worksheet->write(0, 6, 'Average', $format);
$worksheet->write(1, 6, '=AVERAGE(B2:B9)');
```

```
$worksheet->write(0, 7, 'Min', $format);
$worksheet->write(1, 7, '=MIN(B2:B9)');
```

```
$worksheet->write(0, 8, 'Max', $format);
$worksheet->write(1, 8, '=MAX(B2:B9)');
```

```
$worksheet->write(0, 9, 'Standard Deviation',
$format);
$worksheet->write(1, 9, '=STDEV(B2:B9)');
```

```
$worksheet->write(0, 10, 'Kurtosis', $format);
$worksheet->write(1, 10, '=KURT(B2:B9)');
```

```
$workbook->close();
$workbook->sendFile();
```

Vous remarquerez deux choses :

- 1) Il faut un '=' pour démarrer la formule.
- 2) Le nom des cellules dans **la formule** est de type nominatif A1-B6-K9 etc. et non plus de type array 0,1,2,3...

## 6. Zend Framework

L'intégration de la librairie dans un projet Zend\_framework, ne pose aucun problème.

Il faut mettre les classes dans le répertoire /Library de votre projet Zend Framework, et puis de rajouter **Zend\_Loader::registerAutoload()** dans le Bootstrap et à partir de là, plus besoin de définir "require\_once 'Spreadsheet/Excel/Writer.php'".

Une petite subtilité est quand même à noter pour pouvoir générer un fichier Excel dans le framework.

Vous ne pouvez appeler votre code Excel dans une méthode Action, au risque de générer le code Html de la page appelante, parce que la vue se lance automatiquement

Vous devez donc dans une méthode action, appeler une méthode Excel, qui une fois terminer redirigera vers la méthode action demandeuse ....ouf:)

```
<?php
class NouveauController extends Zend_Controller_Action
{

    function IndexAction()
    {
        #faire différents traitements, puis
        appeler la méthode pour le fichier Excel
        $this->Excel;
    }

    private function Excel()
    {
        /**
         * On génère le fichier de base
         */
        $workbook = new
        Spreadsheet_Excel_Writer();
```

```
$workbook->send('base.xls');
$worksheet = $workbook->addWorksheet();

$worksheet->write(1,2, 'toto');

$workbook->close();
$workbook->sendFile();
/**
 * Je renvoie vers l'action qui m'a
appelée.
 */
$this->_redirect($this->IndexAction());
}
}
?>
```

Vous pouvez également empêcher la vue de s'afficher automatiquement avec la méthode suivante.

```
$this->_helper->viewRenderer->setNoRender();
```

## 7. Conclusion

Nous venons d'avoir un rapide coup d'oeil sur une des possibilités de créer des fichiers Excel en PHP.

Je n'ai pas vu toutes les méthodes de la librairie de Xavier Noguer, je vous laisse le soin d'explorer plus en avant la documentation officielle ([Lien20](#)).

J'ai également été confronté à une méthode que je n'ai jamais réussi à implémenter 'insertBitmap()', mais je ne désespère pas :). On peut également noter que j'ai principalement utilisé OpenOffice et Firefox, et que dans le cadre de IE et MS Excel le fichier demande à être enregistré et non pas lu. Il existe d'autres méthodes pour générer des fichiers Excel, cela fera peut-être partie d'un autre article.

Bon travail :)

Retrouvez le tutoriel de Gérard Ernaelsten en ligne : [Lien27](#)

## Les derniers tutoriels et articles

### Les tableaux (XHTML & CSS)

Les tableaux ont été introduits pour pouvoir afficher des données tabulaires. Il n'est plus question d'utiliser des tableaux pour la mise en page d'un site. Nous allons voir dans cet article, les balises se rapportant aux tableaux ainsi que la façon de rendre son tableau plus beau avec du CSS. Une attention particulière sera portée sur l'accessibilité.

#### 1. Introduction

Cet article se base sur un **DOCTYPE XHTML 1.0 Strict**. À la fin de l'article, on aura un tableau qui ressemblera à ceci :

NAVIGATEURS	NOMBRE
Firefox 2.0	34778 (50.7 %)
Internet Explorer 6.0	15176 (22.1 %)
Internet Explorer 7.0	9957 (14.5 %)
Firefox 1.5	3864 (5.6 %)
Opera 9.2	1008 (1.5 %)
Mozilla 5.0	755 (1.1 %)
Firefox 1.0	559 (0.8 %)
Safari 419.3	537 (0.8 %)
Opera 9.1	354 (0.5 %)
Safari 523.1	195 (0.3 %)
Créé par	Adrien Pellegrini PhpMyVisite

#### 2. Les balises

##### 2.1. table

La balise **table** est l'élément de base d'un tableau. Il est interdit d'imbriquer des balises **table**.

```
<table> </table>
```

##### 2.2. tr

La balise **tr** permet de déclarer une ligne du tableau.

```
<table>  
  <tr></tr>  
  <tr></tr>  
</table>
```

##### 2.3. td

La balise **td** permet de créer une cellule dans la ligne du tableau.

```
<table>  
  <tr>  
    <td>Navigateurs</td>  
    <td>Nombre</td>  
  </tr>  
  <tr>  
    <td>Firefox 2.0</td>  
    <td>34778 (50.7 %)</td>  
  </tr>  
  <tr>  
    <td>Internet Explorer 6.0</td>  
    <td>15176 (22.1 %)</td>  
  </tr>  
  <tr>  
    <td>Internet Explorer 7.0</td>
```

```
<td>9957 (14.5 %)</td>  
</tr>  
<tr>  
  <td>Firefox 1.5</td>  
  <td>3864 (5.6 %)</td>  
</tr>  
</table>
```

Voir l'exemple ([Lien28](#))

##### 2.4. th

La balise **th** permet de créer un en-tête dans le tableau. Dans notre exemple **th** définit le titre des colonnes.

```
<table>  
  <tr>  
    <th>Navigateurs</th>  
    <th>Nombre</th>  
  </tr>  
  <tr>  
    <td>Firefox 2.0</td>  
    <td>34778 (50.7 %)</td>  
  </tr>  
  <tr>  
    <td>Internet Explorer 6.0</td>  
    <td>15176 (22.1 %)</td>  
  </tr>  
  <tr>  
    <td>Internet Explorer 7.0</td>  
    <td>9957 (14.5 %)</td>  
  </tr>  
  <tr>  
    <td>Firefox 1.5</td>  
    <td>3864 (5.6 %)</td>  
  </tr>  
</table>
```

Voir l'exemple ([Lien29](#))

##### 2.5. caption

La balise **caption** permet de définir une légende pour le tableau. Cette légende est généralement placée au dessus du tableau. Nous verrons dans la partie CSS le moyen de la déplacer.

```
<table>  
  <caption>Statistique des navigateurs sur l'année  
  2007 de a-pellegrini.developpez.com</caption>  
  <tr>  
    <th>Navigateurs</th>  
    <th>Nombre</th>  
  </tr>  
  <tr>
```

```

<td>Firefox 2.0</td>
<td>34778 (50.7 %)</td>
</tr>
<tr>
<td>Internet Explorer 6.0</td>
<td>15176 (22.1 %)</td>
</tr>
<tr>
<td>Internet Explorer 7.0</td>
<td>9957 (14.5 %)</td>
</tr>
<tr>
<td>Firefox 1.5</td>
<td>3864 (5.6 %)</td>
</tr>
</table>

```

Voir l'exemple ([Lien30](#))

## 2.6. thead, tbody et tfoot

Les balises **thead**, **tbody** et **tfoot** servent respectivement à grouper logiquement des lignes dans un en-tête, un corps et un pied de page.

Le W3C impose que la balise *tfoot* apparaisse avant *tbody*. Si vous ne le faites pas, votre page ne passera pas la validation.

Il est aussi obligatoire de spécifier un *tbody* si vous avez utilisé plus haut un *thead* ou un *tfoot*.

Grouper les lignes de cette façon permet entre autre via CSS de mettre en forme qu'un certain nombre de lignes. Nous verrons un exemple plus tard dans la partie CSS.

```

<table>
  <caption>Statistique des navigateurs sur l'année
  2007 de a-pellegrini.developpez.com</caption>

  <thead>
    <tr>
      <th>Navigateurs</th>
      <th>Nombre</th>
    </tr>
  </thead>

  <tfoot>
    <tr>
      <td>&nbsp;</td>
      <td>Créé par Adrien Pellegrini</td>
    </tr>
  </tfoot>

  <tbody>
    <tr>
      <td>Firefox 2.0</td>
      <td>34778 (50.7 %)</td>
    </tr>
    <tr>
      <td>Internet Explorer 6.0</td>
      <td>15176 (22.1 %)</td>
    </tr>
    <tr>
      <td>Internet Explorer 7.0</td>
      <td>9957 (14.5 %)</td>
    </tr>
    <tr>
      <td>Firefox 1.5</td>
      <td>3864 (5.6 %)</td>
    </tr>
  </tbody>
</table>

```

Voir l'exemple ([Lien31](#))

## 2.7. col et colgroup

Les balises **col** et **colgroup** servent toutes deux à grouper des colonnes afin d'y appliquer un certain style.

Les balises doivent être placées après la balise *table*, après la balise *caption* si elle est renseignée et avant tout le reste.

Il n'est pas nécessaire d'inclure les balises *col* dans une balise *colgroup*. Aussi bien *col* que *colgroup* peut s'utiliser seule.

La balise *col* contient un certain nombre d'attributs, mais un en particulier mérite d'être un peu plus expliqué. Il s'agit de l'attribut *span*. Cet attribut permet de spécifier sur combien de colonnes le style doit être appliqué. Dans l'exemple ci-dessus, il est mis à sa valeur par défaut.

```

<table>
  <caption>Statistique des navigateurs sur l'année
  2007 de a-pellegrini.developpez.com</caption>

  <colgroup>
    <col span="1" width="200" style="background-
    color:#B8C7D3" />
    <col span="1" width="150" style="background-
    color: #CCCCCC" />
  </colgroup>

  <thead>
    <tr>
      <th>Navigateurs</th>
      <th>Nombre</th>
    </tr>
  </thead>

  <tfoot>
    <tr>
      <td>&nbsp;</td>
      <td>Créé par Adrien Pellegrini</td>
    </tr>
  </tfoot>

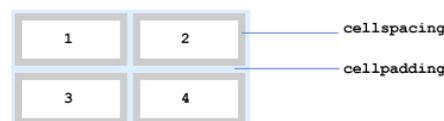
  <tbody>
    <tr>
      <td>Firefox 2.0</td>
      <td>34778 (50.7 %)</td>
    </tr>
    <tr>
      <td>Internet Explorer 6.0</td>
      <td>15176 (22.1 %)</td>
    </tr>
    <tr>
      <td>Internet Explorer 7.0</td>
      <td>9957 (14.5 %)</td>
    </tr>
  </tbody>
</table>

```

Voir l'exemple ([Lien32](#))

## 3. Espacement des cellules

Il est possible grâce à 2 attributs de la balise *table* de spécifier un espacement entre les cellules, **cellspacing** et un espacement entre le contenu et les bords de la cellule, **cellpadding**.



Il est préférable de mettre ces 2 attributs à 0 si vous utilisez une mise en forme CSS.

## 4. Fusion des cellules

### 4.1. colspan

L'attribut **colspan** permet de fusionner plusieurs colonnes.

```
<table>
  <caption>Statistique des navigateurs sur l'année
  2007 de a-pellegrini.developpez.com</caption>

  <colgroup>
    <col span="1" width="200" style="background-
    color:#B8C7D3" />
    <col span="1" width="150" style="background-
    color: #CCCCCC" />
  </colgroup>

  <thead>
    <tr>
      <th>Navigateurs</th>
      <th>Nombre</th>
    </tr>
  </thead>

  <tfoot>
    <tr>
      <td colspan="2">Créé par Adrien
      Pellegrini</td>
    </tr>
  </tfoot>

  <tbody>
    <tr>
      <td>Firefox 2.0</td>
      <td>34778 (50.7 %)</td>
    </tr>
    <tr>
      <td>Internet Explorer 6.0</td>
      <td>15176 (22.1 %)</td>
    </tr>
    <tr>
      <td>Internet Explorer 7.0</td>
      <td>9957 (14.5 %)</td>
    </tr>
  </tbody>
</table>
```

Voir l'exemple ([Lien33](#))

### 4.2. rowspan

L'attribut **rowspan** permet de fusionner plusieurs cellules.

```
<table>
  <caption>Statistique des navigateurs sur l'année
  2007 de a-pellegrini.developpez.com</caption>

  <colgroup>
```

```
<col span="1" width="200" style="background-
color:#B8C7D3" />
  <col span="1" width="150" style="background-
color: #CCCCCC" />
</colgroup>

<thead>
  <tr>
    <th>Navigateurs</th>
    <th>Nombre</th>
  </tr>
</thead>

<tfoot>
  <tr>
    <td rowspan="2">Créé par</td>
    <td>Adrien Pellegrini</td>
  </tr>
  <tr>
    <td>PhpMyVisite</td>
  </tr>
</tfoot>

<tbody>
  <tr>
    <td>Firefox 2.0</td>
    <td>34778 (50.7 %)</td>
  </tr>
  <tr>
    <td>Internet Explorer 6.0</td>
    <td>15176 (22.1 %)</td>
  </tr>
  <tr>
    <td>Internet Explorer 7.0</td>
    <td>9957 (14.5 %)</td>
  </tr>
</tbody>
</table>
```

Voir l'exemple ([Lien34](#))

## 5. Accessibilité

### 5.1. th

Bien que le style de la balise *th* diffère souvent d'une balise *td*, ce n'est son but premier. Pour les programmes utilisés par les malvoyants, cette balise est très importante car elle permet, comme dit précédemment, de lier une cellule à une colonne ou une ligne.

### 5.2. summary

L'attribut **summary** de la balise *table* permet de spécifier une description du tableau beaucoup plus longue que celle de la balise *caption*.

Cela permet à une personne malvoyante de savoir ce que le tableau décrit et si oui ou non il faut s'y intéresser.

L'attribut *summary* au contraire de la balise *caption* n'est pas affiché par les navigateurs habituels.

```
<table summary="Comparaison des statistiques des
différents navigateurs sur la période du 1er janvier
2007
  au 31 décembre 2007 sur le domaine a-
pellegrini.developpez.com">
  <caption>Statistique des navigateurs sur l'année
  2007 de a-pellegrini.developpez.com</caption>

  <colgroup>
```

```

<col span="1" width="200" style="background-
color:#B8C7D3" />
<col span="1" width="150" style="background-
color: #CCCCCC" />
</colgroup>

<thead>
<tr>
<th>Navigateurs</th>
<th>Nombre</th>
</tr>
</thead>

<tfoot>
<tr>
<td rowspan="2">Créé par</td>
<td>Adrien Pellegrini</td>
</tr>
<tr>
<td>PhpMyVisite</td>
</tr>
</tfoot>
<tbody>
<tr>
<td>Firefox 2.0</td>
<td>34778 (50.7 %)</td>
</tr>
<tr>
<td>Internet Explorer 6.0</td>
<td>15176 (22.1 %)</td>
</tr>
<tr>
<td>Internet Explorer 7.0</td>
<td>9957 (14.5 %)</td>
</tr>
</tbody>
</table>

```

Voir l'exemple ([Lien35](#))

### 5.3. abbr

L'attribut **abbr** de la balise *th* sert à spécifier un titre plus court. L'exemple ici n'est pas très bien choisi car il n'est pas nécessaire de mettre un titre plus court.

Pourquoi spécifier un titre plus court ?

Tout simplement pour ne pas faire répéter au navigateur pour malvoyants un titre de colonne trop long à chaque ligne.

```

<table summary="Comparaison des statistiques des
différents navigateurs sur la période du 1er janvier
2007
au 31 décembre 2007 sur le domaine a-
pellegrini.developpez.com">
<caption>Statistique des navigateurs sur l'année
2007 de a-pellegrini.developpez.com</caption>

<colgroup>
<col span="1" width="200" style="background-
color:#B8C7D3" />
<col span="1" width="150" style="background-
color: #CCCCCC" />
</colgroup>
<thead>
<tr>
<th abbr="Nav">Navigateurs</th>
<th abbr="Nb">Nombre</th>
</tr>
</thead>

```

```

<tfoot>
<tr>
<td rowspan="2">Créé par</td>
<td>Adrien Pellegrini</td>
</tr>
<tr>
<td>PhpMyVisite</td>
</tr>
</tfoot>
<tbody>
<tr>
<td>Firefox 2.0</td>
<td>34778 (50.7 %)</td>
</tr>
<tr>
<td>Internet Explorer 6.0</td>
<td>15176 (22.1 %)</td>
</tr>
<tr>
<td>Internet Explorer 7.0</td>
<td>9957 (14.5 %)</td>
</tr>
</tbody>
</table>

```

Voir l'exemple ([Lien36](#))

### 5.4. scope

L'attribut **scope** sert à établir une relation entre l'en-tête (balise *th*) et les cellules de données.

L'attribut *scope* peut prendre 4 valeurs :

1. **col** : la cellule sur laquelle est appliqué le *scope* se rapporte à une colonne
2. **row** : la cellule sur laquelle est appliqué le *scope* se rapporte à une ligne
3. **colgroup** : la cellule sur laquelle est appliqué le *scope* se rapporte à tout le *colgroup*
4. **rowgroup** : la cellule sur laquelle est appliqué le *scope* se rapporte à tout le *rowgroup*

```

<table summary="Comparaison des statistiques des
différents navigateurs sur la période du 1er janvier
2007
au 31 décembre 2007 sur le domaine a-
pellegrini.developpez.com">
<caption>Statistique des navigateurs sur l'année
2007 de a-pellegrini.developpez.com</caption>
<colgroup>
<col span="1" width="200" style="background-
color:#B8C7D3" />
<col span="1" width="150" style="background-
color: #CCCCCC" />
</colgroup>
<thead>
<tr>
<th abbr="Nav" scope="col">Navigateurs</th>
<th abbr="Nb" scope="col">Nombre</th>
</tr>
</thead>
<tfoot>
<tr>
<td rowspan="2">Créé par</td>
<td>Adrien Pellegrini</td>
</tr>
<tr>
<td>PhpMyVisite</td>
</tr>
</tfoot>

```

```

<tbody>
  <tr>
    <td>Firefox 2.0</td>
    <td>34778 (50.7 %)</td>
  </tr>
  <tr>
    <td>Internet Explorer 6.0</td>
    <td>15176 (22.1 %)</td>
  </tr>
  <tr>
    <td>Internet Explorer 7.0</td>
    <td>9957 (14.5 %)</td>
  </tr>
</tbody>
</table>

```

Voir l'exemple ([Lien36](#))

## 5.5. headers

L'attribut **headers** permet de spécifier la liste des en-têtes qui se rapporte à une cellule. Cette liste sera des *id* séparés par un espace.

Cet attribut s'utilise essentiellement quand l'attribut *scope* n'est pas suffisant. Encore une fois ici, l'exemple est assez mal choisi car l'attribut *scope* est largement suffisant.

```

<table summary="Comparaison des statistiques des
différents navigateurs sur la période du 1er janvier
2007
    au 31 décembre 2007 sur le domaine a-
pellegrini.developpez.com">
  <caption>Statistique des navigateurs sur l'année
2007 de a-pellegrini.developpez.com</caption>

  <colgroup>
    <col span="1" width="200" style="background-
color:#B8C7D3" />
    <col span="1" width="150" style="background-
color: #CCCCCC" />
  </colgroup>

  <thead>
    <tr>
      <th id="nav" abbr="Nav"
scope="col">Navigateurs</th>
      <th id="nb" abbr="Nb"
scope="col">Nombre</th>
    </tr>
  </thead>

  <tfoot>
    <tr>
      <td rowspan="2">Créé par</td>
      <td>Adrien Pellegrini</td>
    </tr>
    <tr>
      <td>PhpMyVisite</td>
    </tr>
  </tfoot>

  <tbody>
    <tr>
      <td id="ff2" headers="nav">Firefox 2.0</td>
      <td headers="ff2 nb">34778 (50.7 %)</td>
    </tr>
    <tr>
      <td id="ie6" headers="nav">Internet
Explorer 6.0</td>
      <td headers="ie6 nb">15176 (22.1 %)</td>
    </tr>

```

```

<tr>
  <td id="ie7" headers="nav">Internet
Explorer 7.0</td>
  <td headers="ie7 nb">9957 (14.5 %)</td>
</tr>
</tbody>
</table>

```

Voir l'exemple ([Lien36](#))

## 6. Un peu de CSS

### 6.1. Centrage du tableau

Pour centrer le tableau, il faut faire comme pour centrer tout élément c'est à dire avec un **margin** à **auto**.

Comme vous pourrez le constater, le *caption* ne se centre pas automatiquement avec le tableau. Il faut donc appliquer le même CSS au *caption*.

```

#tab, #tab caption
{
  margin: auto;
}

```

Voir l'exemple ([Lien37](#))

### 6.2. Les bordures

#### 6.2.1. Les bordures externes

Comme à tout élément HTML, il est possible de spécifier une bordure sur l'élément *table*.

Comme vous pourrez le constater également, le *caption* n'est pas pris dans les bordures.

```

#tab
{
  border: #003399 1px solid;
}

```

Voir l'exemple ([Lien38](#))

#### 6.2.2. Les bordures internes

Il existe 2 types de bordures internes. Les bordures dites "fusionnées" et les bordures dites "séparées".

Pour illustrer ces 2 cas il nous faut d'abord mettre des bordures sur les éléments *td* du tableau.

```

#tab td
{
  border: #3399CC 1px solid;
}

```

##### 6.2.2.1. Les bordures séparées

Dans ce cas, les bordures de chaque cellule sont séparées. Pour se faire, il faut utiliser la propriété CSS **border-collapse** à **separate**.

Il est possible de spécifier aussi l'espacement entre les cellules grâce à la propriété CSS **border-spacing** à **x y** ou **x** (ou **x** représente la taille de l'espacement horizontal et **y** le vertical).

```

#tab
{
  border-collapse: separate;
  border-spacing: 10px 5px;
}

```

Voir l'exemple ([Lien39](#))

Les propriétés CSS *border-collapse* et *border-spacing* sont compatibles Netscape, Safari et Mozilla uniquement. Pour que l'espacement soit correct, il faut utiliser les attributs *cellpadding* et *cellspacing* de la balise *table*.

### 6.2.2.2. Les bordures fusionnées

Dans ce cas, les bordures de chaque cellule sont fusionnées. Pour se faire il faut utiliser la propriété CSS **border-collapse** à **collapse**.

```
#tab
{
    border-collapse: collapse;
}
```

Voir l'exemple ([Lien40](#))

### 6.3. Les cellules vides

Il se peut qu'une cellule soit vide (c'est à dire ni de texte ni d'espace insécable ou espace blanc). Le comportement par défaut est d'afficher les cellules vides. Ce comportement peut être changé grâce à la propriété CSS **empty-cells** à **hide**. La propriété CSS prend soit comme valeur *hide*, soit *show*.

Cette propriété CSS est compatible Netscape, Safari et Mozilla uniquement.

```
#tab
{
    empty-cells: hide;
}
```

Voir l'exemple ([Lien41](#))

### 6.4. Emplacement de la légende

Une propriété CSS nous permet de déplacer l'élément *caption*. Cette propriété est **caption-side** qui prend 4 valeurs : *bottom*, *left*, *right*, *top*.

Cette propriété CSS est compatible Netscape, Safari et Mozilla uniquement.

```
#tab
{
    caption-side: bottom;
}
```

Voir l'exemple ([Lien42](#))

### 6.5. Exemple complet

```
#tab, #tab caption
{
    margin: auto;
}
```

```
#tab
{
    border: #DDEEFF 2px solid;
    border-collapse: separate;
    border-spacing: 2px;
    empty-cells: hide;
}

#tab caption
{
    background-color: #DDEEFF;
    font-size: 0.8em;
}

#tab th
{
    color: #996600;
    background-color: #FFCC66;
    border: #FFCC66 1px solid;
    font-variant: small-caps;
    font-size: 0.8em;
    letter-spacing: 1px;
}

#tab td
{
    border: #DDEEFF 1px solid;
    padding-left: 10px;
}

#navcol
{
    width: 200px;
    background-color: #F4FAFD;
}

#numcol
{
    width: 150px;
}

#tab tfoot
{
    font-size: 0.7em;
    background-color: #FFCC66;
    color: #996600;
    letter-spacing: 1px;
}
```

Voir l'exemple ([Lien43](#))

### 7. Conclusion

N'oubliez pas les personnes en difficulté ! Il ne faut pas grand chose de plus à un tableau pour qu'il soit au minimum accessible (ajouter la balise *th*, *caption* et l'attribut *summary*).

Pour la partie CSS libre à votre imagination. Ce n'est pas très compliqué de mettre en forme un tableau et ça demande peu de CSS pour arriver à un beau résultat.

Retrouvez l'article d'Adrien Pellegrini en ligne : [Lien44](#)

# Créer des boutons attrayants et redimensionnables

Ce tutoriel a pour but de vous montrer comment personnaliser vos boutons avec des images et que du CSS.

## 1. Introduction

En matière de réalisation, il existe plusieurs chemins pour arriver au même résultat. La plupart du temps, c'est une question de préférence par rapport à l'élément que vous souhaitez utiliser pour un point précis. J'ai moi aussi des préférences. L'une d'entre elles est de privilégier l'élément **button** par rapport à l'élément **input** de type *submit*. Pourquoi ? Parce que le W3C le dit : *"Les boutons créés avec l'élément **button** fonctionnent de la même manière que ceux créés avec **input**, mais ils offrent de plus riches possibilités au niveau du rendu."*

C'est ce rendu qui m'intéresse. Vous pouvez facilement appliquer des techniques de remplacement d'image sur des boutons pour rendre votre formulaire attractif. La fonctionnalité la plus intéressante est que l'élément **button** peut posséder un contenu. C'est cette possibilité que je vais utiliser dans ce tutoriel.

L'objectif est donc de créer un bouton attractif pouvant gérer des tailles variables afin de ne pas avoir à faire de modifications ultérieures. Nous allons traiter l'élément **button** comme un conteneur et ajouter quelques balises.

## 2. Code (X)HTML

Le voici :

### Code (X)HTML du bouton

```
<button><span><em>Texte du bouton</em></span></button>
```

Ce code est valide et fournit largement de quoi travailler.

Notez que j'utilise deux éléments enfants au lieu d'un car je n'ai pas réussi à me débarrasser d'un remplissage conservé par le bouton. Si l'un d'entre vous réussit à le concevoir avec un seul élément enfant, n'hésitez pas à me contacter :)

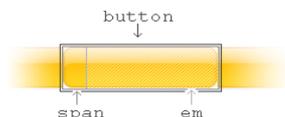
## 3. Le concept

Ce concept vous est probablement familier car il apparaît dans de nombreuses techniques de design. Nous avons une longue image de fond :



Celle-ci fait 550px de large. Je pense que cela devrait suffire pour des boutons :).

Donc, nous utilisons cette image comme arrière-plan d'un élément supérieur (un **span** dans notre cas), le plaçons dans le coin supérieur gauche et ajoutons du remplissage (padding) à gauche. L'élément imbriqué (**em** dans ce cas) possède le même arrière-plan mais il est placé en haut à droite avec un remplissage à droite. Le bouton se redimensionne donc en même temps que le texte.



La hauteur du bouton est fixe dans mon exemple mais vous pouvez utiliser une technique similaire avec quelques balises supplémentaires, et placer le même arrière-plan à chaque coin. Pour s'assurer de l'alignement vertical du texte, j'utilise la

propriété `line-height`.

## 4. Code CSS

### Code de réinitialisation du style du bouton

```
button
{
    border:none;
    background:none;
    padding:0;
    margin:0;
    width:auto;
    overflow:visible;
    text-align:center;
    white-space:nowrap;
    height:40px;
    line-height:38px;
}
```

### Paramétrage des éléments enfants

```
button span, button em
{
    display:block;
    height:40px;
    line-height:38px;
    margin:0;
    color:#954b05;
}
```

Remarquez que la valeur des propriétés **height** et **line-height** sont différentes. C'est à cause de l'ombre de 2px en dessous. La propriété `line-height` centre le texte verticalement, ombre exclue.

### Définition des arrière-plans et remplissages pour les deux éléments enfants

```
button span
{
    padding-left:20px;
    background:url(bg_button.gif) no-repeat 0 0;
}

button em
{
    font-style:normal;
    padding-right:20px;
    background:url(bg_button.gif) no-repeat 100% 0;
}
```

Comme je l'ai mentionné plus haut, il serait plus élégant d'utiliser uniquement **button** et **span**, mais je n'ai pas pu me débarrasser des remplissages du **button**. Si l'un d'entre vous trouve une solution utilisant un seul élément enfant, faites le moi savoir.

Voir le résultat en ligne. ([Lien45](#))

Retrouvez la suite de l'article d'Alen Grakalic traduit par Johann Blais en ligne : [Lien46](#)

### Des applications localisées sous Flex 2

Ce tutoriel vous expliquera comment réaliser des applications localisées, c'est à dire tenant compte de la langue de l'environnement d'exécution.

#### 1. Fonctionnement

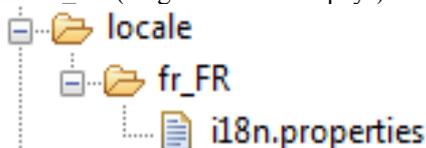
Les ressources permettent de créer des applications localisées à partir du lieu et de la langue de l'environnement d'exécution.

Pour faire une application localisée, il nous faut définir un ou plusieurs fichiers d'extension `.properties` qui contien(nen)t les données à afficher. Ce(s) fichier(s) stocke(nt) les données au format `key = value`, dont voici un exemple :

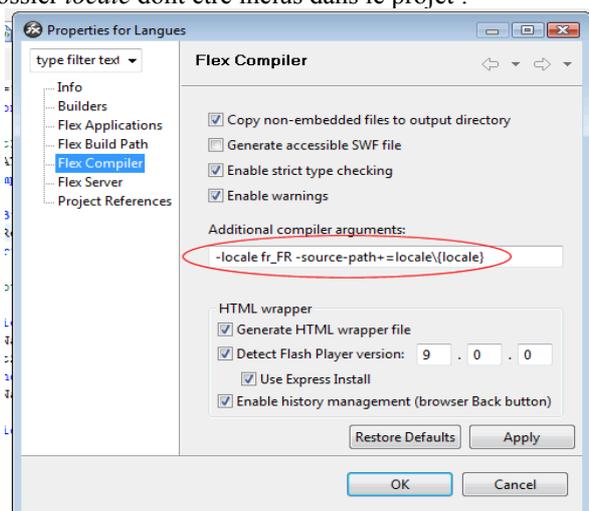
```
// ressources.properties
titleContact=Fiche contact
fieldName=Nom
fieldFName=Prenom
fieldEmail=Courriel
labelBtnSave=Enregistrer
labelBtnCancel=Annuler
```

Ces fichiers seront embarqués dans le fichier SWF résultant de la compilation de votre projet.

Nous placerons ces fichiers ressources dans un dossier nommé *locale* au niveau de chaque projet qui contient lui-même un autre dossier nommé *fr\_FR* (langue underscore pays) :



Ce dossier *locale* doit être inclus dans le projet :



Par défaut, il y a un seul argument **-locale en\_US**, ici nous modifions cet argument en indiquant le choix de la langue et du pays (**fr\_FR**) et nous ajoutons la ligne de commande suivante **-source-path+=locale/{locale}** pour indiquer au compilateur où trouver les fichiers de ressources. Vous aurez compris que la valeur entre accolade correspond à la valeur du premier argument.

Créons le projet Flex ResourceBundleMXML dans lequel nous allons utiliser le metadata *@Resource* pour accéder aux ressources, exemple :

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
xmlns:mx="http://www.adobe.com/2006/mxml"
layout="vertical">

    <mx:Style source="styles.css" />

    <!--
    bundle correspond au nom du fichier de
    ressource
    key le nom de la cle
    -->
    <mx:Panel
title="@Resource(bundle='i18n',key='titleContact') "
width="314" height="183"
layout="absolute"
titleIcon="@Embed('assets/addressbook2.
png')">

        <mx:Form styleName="frmContact"
left="10" right="10" top="10" bottom="10">
            <mx:FormItem
label="@Resource(bundle='i18n',key='fieldName') "
width="100%">
                <mx:TextInput
width="100%" />
            </mx:FormItem>
            <mx:FormItem
label="@Resource(bundle='i18n',key='fieldFName') "
width="100%">
                <mx:TextInput
width="100%" />
            </mx:FormItem>
            <mx:FormItem
label="@Resource(bundle='i18n',key='fieldEmail') "
width="100%">
                <mx:TextInput
width="100%" />
            </mx:FormItem>
        </mx:Form>
        <mx:ControlBar height="42" y="128"
horizontalAlign="right">
            <mx:Button
label="@Resource(bundle='i18n',key='labelBtnSave')"/>
            <mx:Button
label="@Resource(bundle='i18n',key='labelBtnCancel')"/>
        </mx:ControlBar>
    </mx:Panel>
</mx:Application>
```

Nous pouvons réaliser la même chose avec du code Actionscript, en utilisant le metadata *ResourceBundle* et la classe de même nom, ce qui nous donne :

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
xmlns:mx="http://www.adobe.com/2006/mxml"
layout="vertical">

    <mx:Style source="styles.css" />

    <mx:Script>
        <![CDATA[
            import
mx.resources.ResourceBundle;

            [Bindable]
            [ResourceBundle("i18n")]
            private var
i18n:ResourceBundle;

            // Execute apres la creation du
panel
            private function
onCreatePanel():void {
                pnlContact.title =
i18n.getString("titleContact");
            }
        ]]>
    </mx:Script>

    <mx:Panel id="pnlContact" width="314"
height="183" layout="absolute"
titleIcon="@Embed('assets/addressbook2.
png')" creationComplete="onCreatePanel()">
        <mx:Form styleName="frmContact"
left="10" right="10" top="10" bottom="10">
            <mx:FormItem
label="{i18n.getString('fieldName')}" width="100%">
                <mx:TextInput
width="100%" />
            </mx:FormItem>
            <mx:FormItem
label="{i18n.getString('fieldFName')}" width="100%">
                <mx:TextInput
width="100%" />
            </mx:FormItem>
            <mx:FormItem
label="{i18n.getString('fieldEmail')}" width="100%">
                <mx:TextInput
width="100%" />
            </mx:FormItem>
        </mx:Form>
        <mx:ControlBar height="42" y="128"
horizontalAlign="right">
            <mx:Button
label="{i18n.getString('labelBtnSave')}" />
            <mx:Button
label="{i18n.getString('labelBtnCancel')}" />
        </mx:ControlBar>
    </mx:Panel>

</mx:Application>
```

*i18n est une contraction du mot internationalisation*

## 2. Des composants localisés

Nous pouvons donc utiliser la classe *ResourceBundle* pour traduire les composants Flex dans la langue que nous souhaitons. Prenons le cas du composant *DateChooser* (Calendrier). Ce composant propose un ensemble de propriétés nous permettant de modifier les libellés des jours (*dayNames*) et des mois (*monthNames*). Dans un premier temps, créons un fichier de ressources (properties file) :

```
//components.properties
dayNames=D, L, M, M, J, V, S
```

```
monthNames=Janvier, Février, Mars, Avril, Mai, Juin, Juillet,
Août, Septembre, Octobre, Novembre, Décembre
formatString=DD/MM/YYYY
```

Mais ces propriétés à savoir *dayNames* et *monthNames* attendent un tableau, ainsi la méthode *getArrayString()* de *ResourceBundle* permet de retourner un tableau :

```
// Fichier ComponentResourceBundle.mxml
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
xmlns:mx="http://www.adobe.com/2006/mxml"
layout="vertical">
    <mx:Script>
        <![CDATA[
            import
mx.resources.ResourceBundle;

            [Bindable]
            [ResourceBundle("components")]
            private var
i18n:ResourceBundle;
        ]]>
    </mx:Script>
    <mx:DateField
dayNames="{i18n.getStringArray('dayNames')}"
monthNames="{i18n.getStringArray('month
Names')}"
formatString="{i18n.getString('formatSt
ring')}" />
    <mx>DateChooser
dayNames="{i18n.getStringArray('dayNames')}"
monthNames="{i18n.getStringArray('month
Names')}" />
</mx:Application>
```

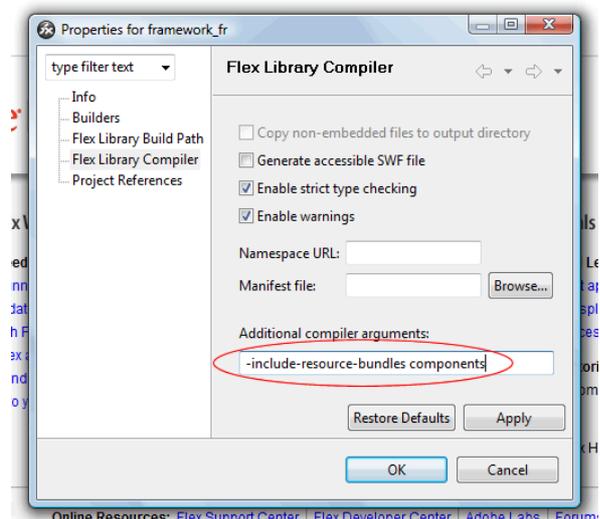
Si vos projets sont le plus souvent destiné, d'un point de vue géographique, à la France, il serait plutôt judicieux de ne pas se préoccuper de traduire à chaque choix les composants. Il suffirait simplement de l'indiquer à travers l'option de commande locale du compilateur en choisissant *fr\_FR*.

C'est pour cette raison que nous allons créer un composant compilé (swc) qui contiendra tous les fichiers de ressources.

Créons un projet de librairie flex que nous appellerons *framework\_rb* et plaçons-y le fichier de ressources précédent. Nous devons indiquer au projet, le(s) fichier(s) de ressources à inclure, ceci avec la ligne de commande suivante :

```
-include-resource-bundles nomDuFichierProperties
```

Dans notre cas :



Note: le nom du fichier de ressources sans son extension.  
Nous obtenons le fichier `framework_rb.swc` dans le dossier `bin`. Il nous faut maintenant ajouter ce composant à notre projet `ComponentResourceBundle`.

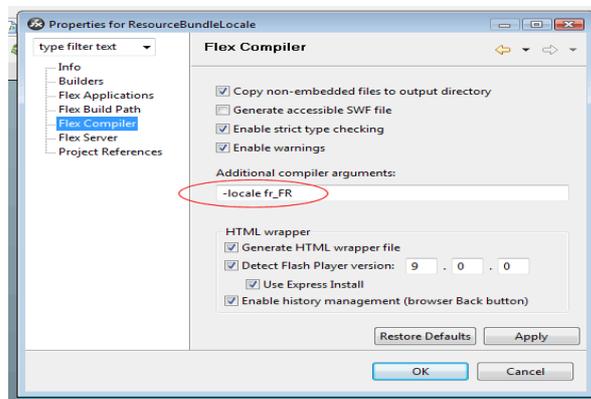
### 3. Encore plus de facilité

Il existe dans le dossier suivant :

```
C:\Program Files\Adobe\Flex Builder 2\Flex SDK  
2\frameworks\locale\en_US
```

plusieurs fichiers de ressources qui contiennent les propriétés par défaut pour tous les composants Flex. Il vous suffit de créer un projet de librairie Flex comme précédemment et d'y inclure ces fichiers traduits dans la langue de votre choix. Après avoir obtenu le fichier `compiler (.swc)`, créer un dossier `fr_FR` dans le dossier indiqué au-dessus et ajoutez-y le composant compilé (`swc`).

Maintenant pour tous vos projets à venir, il vous suffira simplement de modifier l'option de commande suivante par le nom du dossier de votre choix de langue :



Dans la version 3 de Flex, nous pouvons travailler avec les ressources de manière dynamique et ainsi permettre à l'utilisateur de choisir la langue de l'application à l'exécution. J'en parlerai dans un prochain tutoriel

Retrouvez l'article d'Olivier Bugalotto en ligne : [Lien47](#)

## Compiler des projets Flex en ligne de commande

Ce tutoriel vous expliquera comment compiler vos projets Flex en ligne de commande. Cette méthode vous permettra d'augmenter votre productivité tout en vous affranchissant de tout environnement de développement.

### 1. Introduction

Vous ne vous êtes jamais dit que compiler vos projets Flex ou AIR sans vous reposer sur Flex Builder ou d'autre IDE serait plutôt pas mal ? Oui je vous vois venir, ça ne sert à rien, et pourtant !)

Cela permet de comprendre ce qui se passe dans les coulisses, et surtout, une fois maîtrisé, vous avez bien plus de contrôles sur tout ce que vous faites. C'est également une étape nécessaire pour flexouiller sans Flex Builder.

Bon bon. Oui ça peut paraître rébarbatif au début. Mais ça reste simple pour compiler un simple `.as` ou un `MXML`. Pour compiler tout un projet, ou un `framework`, cela demande un peu plus d'organisation, mais rien de bien compliqué.

Toutes les sources utilisées dans ce tutorial sont ici ([Lien48](#)).

C'est parti !)

### 2. Exploration ...

Un petit tour d'horizon de nos compilateurs déjà. Ils se trouvent dans le dossier `{pathFlexSDK3}\bin\`. Ceux qui nous intéressent ici sont :

- Le compilateur `mxmlc.exe` (ainsi que le batch et le shell `amxmlc`, qui configure `mxmlc` pour compiler du AIR)
- Le compilateur `compc.exe` (ainsi que le batch et le shell `acompc`, qui configure `compc` pour compiler une bibliothèque contenant des classes de AIR)
- Le compilateur `asdoc.exe`, qui permet de générer de la documentation.

Dans cet article, on ne verra que la compilation avec `mxmlc`. Des articles ultérieurs expliqueront comment compiler un ensemble de classes en une bibliothèque `SWC`, générer une documentation avec l'`ASDOC`, et compiler avec `style` vos applications AIR. Que du bonheur en perspective !

### 3. Compilation simple ...

On supposera, dans la suite, qu'on se trouve dans un environnement Windows (des fichiers batch seront utilisés, il suffira de les transposer en shell pour les linuxiens).

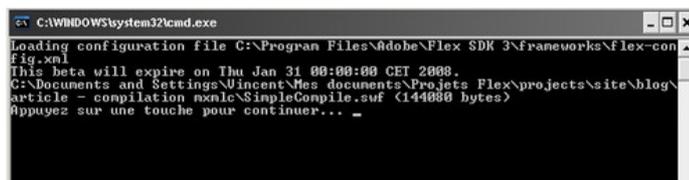
Il faut garder à l'esprit qu'on cherche à être productif, nous allons

donc utiliser des fichiers batch afin de faciliter l'appel des commandes.

Pour expérimenter, compilons un `mxml` simple. Le fichier batch à créer – dans le même répertoire pour cet exemple – nécessite de créer une variable mémorisant le chemin absolu ou relatif vers le compilateur `mxmlc` (préférez des chemins absolus pour la réutilisabilité).

```
@echo off  
  
REM the path of the mxmlc compiler  
  
SET mxmlcPath="C:\Program Files\Adobe\Flex SDK  
3\bin\mxmlc.exe"  
  
REM The mxml to compile  
  
SET mxmlToCompile="SimpleCompile.mxml"  
  
%mxmlcPath% %mxmlToCompile%  
  
pause
```

On l'exécute...



Très bien, nous avons notre `swf` créé juste à côté de notre `mxml` et de son batch.

C'est bien pratique pour compiler simplement un petit `mxml` qu'on fait en test par exemple. Solution à ne pas comparer avec la version artillerie lourde, du `style`, Flex Builder -> New Project, et hop on se retrouve avec pas mal de fichiers pour... juste un test...

Ainsi, utiliser un simple batch a l'avantage d'être facile à mettre en

place, et donc pratique pour des travaux de petite envergure.

#### 4. Compilation à l'aide d'un XML...

Si on veut aller plus loin, la méthode précédente ne suffit plus.

En effet, le compilateur prend en charge un nombre important d'options. La liste complète de ces options se trouve dans les sources jointes à ce tutoriel.

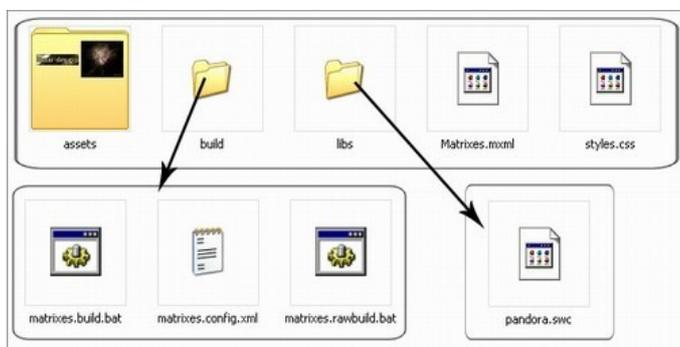
Dès lors, le batch n'est plus vraiment adapté à contenir toutes ces options. Au niveau maintenance et productivité, ce n'est plus gérable de créer une variable pour chaque option du compilateur, ou d'avoir une ligne de commandes inbuables. Nous allons donc utiliser un xml de configuration, qui sera spécifique à chaque projet, dans lequel les options du compilateur seront mémorisées.

Avant de se lancer, quelques rappels et conseils pratiques.

La bonne habitude à prendre dès lors qu'on travaille en POO est d'organiser son travail sous forme de packages. Notre projet a une racine, contenant nos packages, et d'autres éléments. Ce n'est normalement pas nouveau pour vous (j'espère). La bonne pratique est d'ajouter à cette racine d'autres dossiers types, comme par exemple un dossier 'docs', contenant notre documentation, un dossier 'bin' ou 'deploy' pour notre sortie, etc..

Une bonne idée est de faire un dossier 'build', qui contiendra tous nos fichiers de configuration pour compiler notre projet, et d'ajouter des raccourcis des batch où bon vous semble, si vous jugez que cela vous fait gagner du temps.

Je vais prendre comme exemple une mini-application que j'ai réalisé il y a quelques temps, et qui illustre l'utilisation de la classe Matrix... Un truc bateau en somme, qui n'utilise qu'un simple MXML et une bibliothèque (d'une seule classe, pour l'exemple).



On désire compiler Matrixes.mxml, mais avec certaines options. Parmi elles, le nom du swf produit, les source-path, des library à inclure, le frame-rate, des metadata, etc.

Notre batch, avec les options du compilateur devient :

```
@echo off
REM the path of the mxmllc compiler
SET mxmllcPath="C:\Program Files\Adobe\Flex SDK
3\bin\mxmllc.exe"
%mxmllcPath% -sp ../ -include-libraries
../libs/pandora.swc -use-network=false -default-size
1024 768 -default-frame-rate 25
-default-background-color #FFFFFF -o
../matrixes_BasicBuild.swf -file-specs ../Matrixes.mxml
pause
```

Assez rébarbatif n'est-ce pas ?

Voyons l'équivalent avec le xml. Voici celui qui va paramétrer les options de notre application, matrixes.config.xml :

```
<flex-config>
    <!--Compiler options -->
    <compiler>
        <source-path append="true">
<path-element>../</path-element>
        </source-path>
<library-path append="true">
<path-element>../libs/pandora.swc</path-element>
        </library-path>
        <debug>true</debug>
    </compiler>
    <!--Files to compile -->
    <file-specs>
<path-element>../Matrixes.mxml</path-element>
    </file-specs>
    <!--Output file -->
    <o>../matrixes.swf</o>
    <!--Other params -->
    <default-background-color>#FFFFFF</default-
background-color>
    <default-frame-rate>25</default-frame-rate>
    <default-size>
        <width>1024</width>
        <height>768</height>
    </default-size>
    <!-- Enables SWFs to access the network. -->
    <use-network>false</use-network>
</flex-config>
```

C'est déjà mieux. Certes plus 'bavard', mais bien mieux organisé. Pour savoir comment obtenir le nom de la balise qui va se relier à la bonne option, il faut respecter toujours la même logique. Si on regarde la doc des options du compilateur (lien de téléchargement donné plus haut), nous avons par exemple : -compiler.source-path [path-element], d'alias -sp. Le raisonnement est le même que pour des packages. -source-path est une commande de compiler. Ainsi, nous le regroupons dans le XML dans une balise . En revanche, -o est l'alias de la commande top-level 'output'. La balise est donc à placer directement sous la racine. La logique est la même pour les autres commandes.

Un point important à noter est que les chemins relatifs spécifiés sont pris par rapport à l'emplacement du xml. Une autre remarque : j'ai pu constater que la commande -l, (library-path dans le xml) utilisée en dehors du xml de configuration semble neutraliser l'import des bibliothèques chargés par le fichier de configuration par défaut (flex-config.xml). En revanche, utilisée dans un autre xml de configuration, les bibliothèques s'ajoutent normalement aux autres sans les neutraliser. Utilisez donc -include-libraries en ligne de commande, qui ne produit pas cette erreur. En passant, la différence entre ces deux commandes : -include-libraries inclut dans le swf la totalité de la bibliothèque, alors que -l (library-path) inclut uniquement les classes requises par l'application.

Maintenant, nous devons modifier le batch pour qu'il charge notre fichier de configuration.

Pour cela, nous utilisons la commande -load-config.

```
@echo off

REM the path of the mxmmlc compiler

SET mxmmlcPath="C:\Program Files\Adobe\Flex SDK
3\bin\mxmmlc.exe"

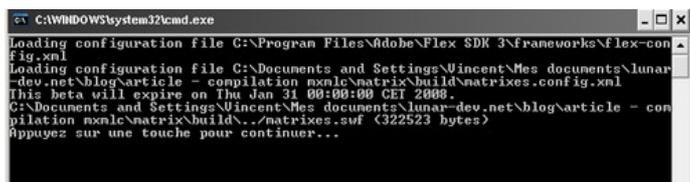
REM the path of your xml configuration

SET xmlConfigPath=matrices.config.xml

%mxmmlcPath% -load-config+%xmlConfigPath%

pause
```

Il n'y a plus qu'à lancer le batch.



Remarquez la ligne supplémentaire indiquant que le compilateur a bien chargé notre xml de configuration (en plus du xml de configuration de flex par défaut).

Pour les développeurs AIR, tout ceci est valable, à ceci près qu'il faut invoquer amxmlc.bat au lieu de mxmmlc.exe. amxmlc charge air-config.xml au lieu de flex-config.xml afin d'inclure les classes propres à AIR. J'en dirai plus sur la compilation de projets AIR dans un autre article.

### 5. Vers l'automatisation ...

Réécrire un xml de configuration à chaque projet serait fastidieux. J'utilise donc un template de xml qui convient pour 95% des projets. Je l'accompagne aussi d'un template de batch, où seule une ligne est à modifier. On gagne un temps précieux, puisqu'il suffit à tout mettre en place.

De plus, on voit vite un avantage se profiler : la portabilité du xml et son implémentation dans bon nombre de langages (Java, PHP, etc.), offrent un bon panel de possibilités de gestion. Pourquoi pas une petite appli en AIR qui créerait automatiquement vos xml de configuration par exemple ?

C'est tout pour cette fois-ci ! Toute remarque est la bienvenue.

### 6. Conclusion

Dans cet article, nous avons vu comment compiler des projets Flex à l'aide des compilateurs en ligne de commande. Nous avons pu approfondir en mettant en place une méthode afin d'améliorer la rapidité de mise en service et notre productivité, tout cela sans l'utilisation d'environnements de développement. Une étape indispensable pour tout développeur Flex et Air voulant s'orienter vers des outils libres.

Retrouvez l'article de Vincent Petithory en ligne : [Lien49](#)

## Les derniers tutoriels et articles

### Introduction aux contrôles templates pour Asp.Net 2.0 en C#

Cet article constitue une introduction à la création de contrôles templates pour Asp.Net en C#. Il se présentera sous la forme d'un tutoriel pour créer un bouton paramétrable, de style "Veuillez patienter ..."

#### 1. Introduction

A travers ce tutoriel, nous allons expliquer les fondements de la création d'un contrôle template pour **Asp.Net 2.0**. et nous allons les utiliser pour créer un contrôle du type "bouton veuillez patienter". Tout au long de cet article, je vais utiliser **Visual C# 2008**.

D'une manière générale, les contrôles serveur d'Asp.Net ont une apparence par défaut, paramétrables grâce à des propriétés ou des feuilles de styles.

Un contrôle template (ou contrôle modèle) offre une complète personnalisation de son apparence en permettant de spécifier un ensemble d'éléments HTML ou de contrôles serveur qui lui servira de rendu.

#### 2. Présentation du contrôle type "Bouton veuillez patienter"

Lors d'un post-pack, il peut arriver qu'on ait des traitements côté serveur plus ou moins long à effectuer, comme une validation de formulaire via un web-services, un ensemble de tâches pour un paramétrage, ... Bref, une tâche assez longue qui peut perturber un utilisateur qui verrait son navigateur bloqué, sans page de confirmation créée rapidement.

Ce que j'appelle un bouton "veuillez patienter" est un bouton qui effectue ce postback et qui affiche un message pour informer l'utilisateur que la tâche est un peu longue, et qu'il peut aller se chercher un café.

Après avoir un peu cherché sur le net dans les différents tutoriels qui traitent de ce sujet, j'ai souvent vu des articles qui présentaient la réécriture complète d'un bouton, avec une propriété paramétrable qui affiche un texte dans un label très moche, ce que je trouve somme toute très limité.

Ce dont j'aurai besoin, c'est d'un contrôle où je maîtrise complètement le rendu, en fonction des conditions inhérentes à la page du moment.

Ce qu'il me faut, c'est la possibilité d'afficher un bouton, et lorsque je clique dessus apparaisse un rendu html que j'aurai préalablement décrit dans ce contrôle.

De dire : ah ba tiens, pour cette page, je veux afficher "veuillez patienter" en rouge sur fond gris.

Et puis pouvoir réutiliser plus loin ce contrôle en affichant cette fois-ci "attention, nous vérifions les informations saisies, cela peut durer de 1 à 5 minutes", dans un cadre vert aux bordures jaunes, couleur bleue sur fond gris.

Bref, vous l'aurez compris ... : la présentation que je veux, où je veux.

C'est ici qu'interviennent les **contrôles templates**.

#### 3. Les contrôles templates

##### 3.1. L'exemple du Repeater

Appelés aussi contrôles "modèles", ce sont des contrôles où l'on a la possibilité de définir nous même le rendu visuel du contrôle.

Un contrôle très connu de contrôle template est le repeater ([Lien100](#)). Il s'agit d'un contrôle de liste lié aux données permettant une présentation personnalisée, grâce à l'application d'un modèle spécifié à chacun des éléments figurant dans la liste. Son utilisation par exemple, permet d'afficher une liste de données (implémentant IEnumerable) dans notre modèle :

```
<asp:Repeater id="Repeater1" runat="server" DataSource='<
%#new string[] { "Nico-pyright", "Cardi", "Dev01", "..."} %>'>
    <HeaderTemplate>
        <div style="border:solid 1px black">Liste des
utilisateurs :
    </HeaderTemplate>
    <ItemTemplate>
        <div style="padding-left:10px"><%#
Container.DataItem%></div>
    </ItemTemplate>
    <FooterTemplate>
        </div>
    </FooterTemplate>
</asp:Repeater>
```

Dans cet exemple de repeater, on voit bien qu'on est maître de la présentation visuelle du tableau de chaîne que l'on a bindé au repeater. Le repeater fournit la fonctionnalité de "répéter" suivant une source de données, et nous fournissons la présentation.

Ce qu'on veut faire pour notre bouton "veuillez patienter" suit exactement le même principe. Une fonctionnalité définie, immuable : à savoir la fonctionnalité d'un bouton. Et une présentation qui reste au bon vouloir de l'utilisateur.

Notez bien qu'il y a une différence entre un contrôle template qui permet de définir complètement notre rendu visuel et l'utilisation d'une feuille de styles, qui permet d'adapter légèrement la mise en page d'un contrôle.

Par exemple, l'utilisation de styles va permettre de modifier la propriété **BackColor** d'un contrôle, tandis qu'un template va nous permettre de rajouter une table html, une présentation complexe, ou d'autres contrôles.

##### 3.2. Créer son propre contrôle

Nous avons remarqué dans l'exemple du Repeater, qu'il est possible d'intervenir à plusieurs niveaux, grâce aux trois templates ci-dessus (il y en a d'autres pour le repeater).

Il s'agit des HeaderTemplate, ItemTemplate et FooterTemplate. La fonctionnalité principale d'un contrôle template réside dans l'utilisation de ces différents templates.

Chacun de ces templates est de type **ITemplate**.

Cette interface définit la méthode **InstantiateIn**.

ASP.NET va parser le contenu de la balise template de notre contrôle et va appeler la méthode **InstantiateIn** à chaque template trouvé. Le contrôle crée à chaque fois l'arbre des contrôles que représente le contenu du template.

Tout le principe réside dans cette implémentation.

Lors de l'appel à **CreateChildControls**, on va utiliser le template pour l'ajouter à l'arbre des contrôles de notre contrôle, grâce à la méthode **InstantiateIn**.

Il sera également possible de définir un rendu par défaut si aucun template n'est utilisé.

Nous allons voir ce fonctionnement en détail dans l'exemple qui suit : la création d'un contrôle de type "bouton veuillez patienter".

#### 4. Création du contrôle bouton veuillez patienter

L'objectif est d'avoir un contrôle qui puisse se définir par exemple ainsi :

```
<Test:PleaseWaitButton ID="PWB" runat="server">
  <PleaseWaitMessageTemplate>
    <p style="background-color:
#eaf2d9;border:1px solid #cccc99;color:green;width:500px;text-align:center">
      Veuillez patienter quelques instants,
    <br/>
    nous vérifions les informations que
    vous avez saisies ...<br />
    <asp:Image runat="server"
ImageUrl="wait.gif" />
  </p>
</PleaseWaitMessageTemplate>
</Test:PleaseWaitButton>
```

Et qui lors d'un click puisse afficher quelque chose comme ça :



#### 4.1. Création du contrôle

Ici notre contrôle s'appelle **PleaseWaitButton**. On lui définit son rendu visuel dans le template **PleaseWaitMessageTemplate**.

En l'occurrence, on affiche un texte pour patienter, et une image style barre de progression infinie.

Nous allons donc créer une classe qui va hériter de **Control**. Cette classe implémentera aussi **INamingContainer**.

Sachez simplement que par l'intermédiaire de cette interface, on va pouvoir identifier les contrôles en fournissant un espace de noms à tous les contrôles serveurs qu'il contient.

```
[ParseChildren(true)]
[DefaultProperty("Text")]
public class PleaseWaitButton : Control, INamingContainer
{
}
```

La classe doit avoir l'attribut **ParseChildren(true)** (hérité si on dérive de **WebControl**) afin de dire au parseur d'interpréter le contenu en tant que propriété et non en tant que contrôles enfants.

Cette classe contiendra bien entendu une propriété template, de type **ITemplate**. L'attribut **TemplateContainer** permettra de savoir quel type de contrôle sera utilisé lors d'un databinding du genre :

```
<%# Container.Item %>
```

Non utilisée dans cet exemple, cet attribut est ici à titre explicatif. L'attribut **PersistenceMode** indique que la propriété persiste dans le contrôle serveur ASP.NET en tant que balise imbriquée.

```
private ITemplate _pleaseWaitMessageTemplate = null;
```

```
[Browsable(false), DefaultValue(null),
PersistenceMode(PersistenceMode.InnerProperty),
TemplateContainer(typeof(TemplateItem))]
public ITemplate PleaseWaitMessageTemplate
{
    get { return _pleaseWaitMessageTemplate; }
    set { _pleaseWaitMessageTemplate = value; }
}
```

C'est la méthode **CreateChildControls** qui contient toute la logique de la création du contrôle template.

On commence par créer un **Panel** caché, auquel on ajoute le template (chargé et parsé grâce à **InstantiateIn**) s'il est défini, ou un contrôle littéral avec un texte par défaut s'il ne l'est pas.

Ensuite, on ajoute le bouton, auquel on affecte la propriété **Text** et on associe un handler de click.

On ajoute tout ça à la collection de contrôles du contrôle template. Notez que le template instancié (**TemplateItem**) est une classe qui implémente **INamingContainer** et qui doit dériver de **Control**, directement ou indirectement. Il est assez classique de faire dériver notre élément de template de la classe **Panel**. Ici, nous dériverons directement de **Control**.

```
[ToolboxItem(false)]
public class TemplateItem : Control, INamingContainer
{
}

protected override void CreateChildControls()
{
    Controls.Clear();
    // On crée un panel caché
    Panel panelMessage = new Panel();
    panelMessage.ID = "_panel";
    panelMessage.Attributes["style"] = "display:none";
    if (PleaseWaitMessageTemplate != null)
    {
        // si un template est défini, on l'utilise
        TemplateItem templateItem = new
        TemplateItem();

        PleaseWaitMessageTemplate.InstantiateIn(templateItem);
        panelMessage.Controls.Add(templateItem);
    }
    else
    {

```

```

        // sinon, on crée un message par défaut tout
moche
        panelMessage.Controls.Add(new
LiteralControl("Veuillez patienter ..."));
    }
    // on crée le bouton avec la propriété Text ou la
chaine "OK" si elle n'est pas définie
    // et on associe un handler de click
    Button boutonValidation = new Button();
    boutonValidation.ID = "_button";
    boutonValidation.Text = Text;
    boutonValidation.Click += b_Click;

    // on ajoute le panel et le bouton
    Controls.Add(panelMessage);
    Controls.Add(boutonValidation);
}

```

Pour fonctionner comme un bouton classique, notre contrôle va publier une propriété Text qui aura également une valeur par défaut, car je n'aime pas les boutons sans texte.

On pourra également associer une méthode pour un traitement particulier lors du click sur le bouton.

Notre contrôle devra également s'insérer correctement dans un cycle de validation de la page asp.net. Ainsi, nous gérons la validation et nous exposerons une propriété ValidationGroup.

```

[Bindable(true), Category("Behavior"), Description("Le groupe
de validation")]

```

```

public string ValidationGroup
{
    get { return (string)ViewState["ValidationGroup"] ??
string.Empty; }
    set { ViewState["ValidationGroup"] = value; }
}

```

```

[Bindable(true), Category("Appearance"), DefaultValue("OK"),
Description("Le texte du bouton")]

```

```

public string Text
{
    get { return (string)ViewState["Text"] ?? "OK"; }
    set { ViewState["Text"] = value; }
}

```

```

private event EventHandler _clickHandler;
public event EventHandler Click
{
    add { _clickHandler += value; }
    remove { _clickHandler -= value; }
}

```

Pour pouvoir parcourir les sous-contrôles de ce contrôle, il faudra surcharger la propriété **Controls**. Nous appellerons la méthode **EnsureChildControls** qui appelle **CreateChildControls** si cela n'a pas déjà été fait.

```

public override ControlCollection Controls
{
    get { EnsureChildControls(); return base.Controls; }
}

```

Lors du click sur le bouton, nous allons dans un premier temps demander la validation de la page puis déclencher l'événement de l'utilisateur, s'il est défini.

```

private void b_Click(object sender, EventArgs e)
{
    // on déclenche la validation manuelle
    if (!string.IsNullOrEmpty(ValidationGroup))
        Page.Validate(ValidationGroup);
    else
        Page.Validate();
    // on lève l'événement du click, s'il y en a un
    if (_clickHandler != null)
        _clickHandler(sender, e);
}

```

La validation est une étape cruciale du cycle de vie de la page asp.net. Notre contrôle se doit de la respecter. Pour demander l'évaluation de cette validation, nous avons besoin d'appeler explicitement la méthode **Page.Validate()**.

Si le bouton fait parti d'un groupe de validation, l'appel à **Page.Validate** devra passer en paramètre le nom du groupe de validation. **Page.Validate()** est une validation côté serveur, c'est à dire sensible à la méthode **OnServerValidate** d'un **CustomValidator** par exemple.

Il est également important de prendre en charge la validation côté client, pour des validators comme le **RequiredFieldValidator**. Pour ce faire, nous devons appeler explicitement côté client une méthode : **Page\_ClientValidate()**.

C'est également à ce moment là que nous devons gérer la visibilité de notre message pour patienter.

Pour ce faire, nous allons définir une méthode javascript que nous allons appeler par l'intermédiaire de la propriété **OnClientClick** du bouton.

Mais avant toute chose, nous avons besoin d'affecter un ID à notre panel, pour pouvoir le manipuler côté javascript :

```

protected override void OnPreRender(EventArgs e)
{
    // on cherche le panel pour récupérer son ClientID
    // on cherche le bouton pour utiliser le ClientID du panel
dans une fonction javascript associée à la propriété OnClientClick
    // cette opération ne peut pas être faite dans le
CreateChildControls, car cela serait trop tôt
    Panel panel = (Panel)FindControl("_panel");
    Button bouton = (Button)FindControl("_button");
    bouton.OnClientClick =
string.Format("checkForm('{0}')" , panel.ClientID);
    base.OnPreRender(e);
}

```

On commence donc par rechercher le panel et on lui affecte un ID. Ensuite, il ne reste plus qu'à rechercher le bouton et à affecter la propriété **OnClientClick** avec le **ClientID** du panel.

Notez qu'on ne peut pas faire ce traitement dans le **CreateChildControls** car l'utilisation du **ClientID** trop tôt provoquerait des erreurs d'identification des contrôles.

La validation côté client et les affichages/masquages du message pour patienter nécessitent du javascript.

Nous allons donc utiliser **Page.ClientScript.RegisterStartupScript** pour enregistrer notre script et nos méthodes javascript.

Le principe est le suivant :

- La méthode `checkForm` est appelée lors du click client sur le bouton.
- On appelle la validation côté client grâce à `Page_ClientValidate` (si le bouton fait partie d'un groupe de validation, on devra passer ce groupe en paramètre). On interceptera une erreur si jamais il n'y a aucune validation.
- Si la validation est bonne, on affiche le panel (représenté par un div) et donc le message pour patienter. Si elle n'est pas bonne, on le masque.

```
protected override void Render(HtmlTextWriter writer)
{
    // création des scripts côté client :
    // si il y a un groupe de validation, on appelle la fonction
    // de validation client avec le groupe en paramètre, sinon sans
    // paramètre
    string validationGroupParameters =
    string.IsNullOrEmpty(ValidationGroup) ? string.Empty :
    string.Format("{0}", ValidationGroup);

    // si la validation est OK, on affiche le panel (et donc le
    // message d'attente)
    // si elle n'est pas OK, on le cache et on retourne faux
    string script = @"function getObj(id)
{
    var o;
    if (document.getElementById)
    {
        o = document.getElementById(id).style;
    }
    else if (document.layers)
    {
        o = document.layers[id];
    }
    else if (document.all)
    {
        o = document.all[id].style;
    }
    return o;
}
function setDisplay(id)
{
    var o = getObj(id);
    if (o)
    {
        o.display = 'block';
    }
}

function unsetDisplay(id)
{
    var o = getObj(id);
    if (o)
    {
        o.display = 'none';
    }
}";
```

```
}
function checkForm(divWaiting)
{
    try
    {
        if (!Page_ClientValidate(" + validationGroupParameters +
        @""))
        {
            unsetDisplay(divWaiting);
            return false;
        }
    }
    catch (e) {}
    setDisplay(divWaiting);
};

Page.ClientScript.RegisterStartupScript(GetType(),
"javascriptButton", script, true);

base.Render(writer);
}
```

## 4.2.Exemples d'utilisation

Nous aurons 2 cas de figures, avec ou sans validation.

### 4.2.1.Sans validation

Il suffira d'utiliser le contrôle de cette façon par exemple :

```
<Test:PleaseWaitButton ID="PWB" runat="server"
OnClick="ClickButton">
    <PleaseWaitMessageTemplate>
        <p style="background-color:
#eaf2d9;border: 1px solid #cccc99;color:green;width:500px;text-
align:center">
            Veuillez patienter quelques instants,
<br/>
            nous vérifions les informations que
vous avez saisies ...<br />
            <asp:Image runat="server"
ImageUrl="wait.gif" />
        </p>
    </PleaseWaitMessageTemplate>
</Test:PleaseWaitButton>
```

Avec dans le code behind, quelque chose comme ca pour simuler un long traitement.

```
protected void ClickButton(object sender, EventArgs e)
{
    Thread.Sleep(2000);
}
```

### 4.2.1.Avec validation

Pour les exemples de validation, j'utiliserai un `RequiredFieldValidator` et un `CustomValidator`.

#### 4.2.1.1.Validation sans groupe

On définit nos validators sans groupe de validation, le contrôle template reste inchangé :

```

<asp:TextBox runat="server" ID="LeTextBox" />
<asp:RequiredFieldValidator runat="server"
ControlToValidate="LeTextBox"
ErrorMessage="Le champ doit être renseigné"
Display="dynamic" />
<asp:CustomValidator runat="server"
OnServerValidate="ValidateFunction" Display="dynamic"
ErrorMessage="Le champ doit être égal à ABC"
/>

<Test:PleaseWaitButton ID="PWB" runat="server"
OnClick="ClickButton">
    <PleaseWaitMessageTemplate>
        <p style="background-color:
#eaf2d9;border:1px solid #cccc99;color:green;width:500px;text-
align:center">
                Veuillez patienter quelques instants,
<br/>
                nous vérifions les informations que
vous avez saisies ...<br />
        <asp:Image runat="server"
ImageUrl="wait.gif" />
        </p>
    </PleaseWaitMessageTemplate>
</Test:PleaseWaitButton>

```

Et dans le code behind :

```

protected void ClickButton(object sender, EventArgs e)
{
    if (Page.IsValid)
    {
        Thread.Sleep(2000);
        Response.Write("<br/>Informations OK<br/>");
    }
}

protected void ValidateFunction(object source,
ServerValidateEventArgs args)
{
    args.IsValid = LeTextBox.Text == "ABC";
}

```

#### 4.2.1.2.Validation avec groupe de validation

Pour la validation avec un groupe de validation, cela fonctionne de la même façon, on renseigne juste les propriétés ValidationGroup :

```

<asp:TextBox runat="server" ID="LeTextBox" />
<asp:RequiredFieldValidator runat="server"
ControlToValidate="LeTextBox"
ErrorMessage="Le champ doit être renseigné"
Display="dynamic" ValidationGroup="g" />
<asp:CustomValidator runat="server"
OnServerValidate="ValidateFunction" Display="dynamic"
ErrorMessage="Le champ doit être égal à ABC"
ValidationGroup="g" />

<Test:PleaseWaitButton ID="PWB" runat="server"
OnClick="ClickButton" ValidationGroup="g">
    <PleaseWaitMessageTemplate>
        <p style="background-color:
#eaf2d9;border:1px solid #cccc99;color:green;width:500px;text-
align:center">
                Veuillez patienter quelques instants,
<br/>
                nous vérifions les informations que
vous avez saisies ...<br />
        <asp:Image runat="server"
ImageUrl="wait.gif" />
        </p>
    </PleaseWaitMessageTemplate>
</Test:PleaseWaitButton>

```

#### 5.Conclusion

Grâce à ce tutorial, nous avons vu une introduction à la création de contrôle template. Nous l'avons ensuite appliqué pour la réalisation d'un contrôle de type "bouton veuillez patienter". Nous avons vu également les éléments primordiaux qui permettent d'assurer la phase de validation de notre contrôle, côté serveur et côté client.

J'espère que vous avez trouvé cet article intéressant et que le contrôle template pourra vous être utile. Ne négligez pas la validation qui est un point critique mais terriblement utile. Connaître et tirer parti des mécanismes de validation peut être un atout non négligeable.

Retrouvez l'article de Nico-pyright en ligne : [Lien101](#)

## Les livres .Net

### Linq in Action

#### Critique du livre par Thomas Lebrun

Je dois admettre que je suis très impressionné par ce livre. Non pas par la le niveau technique du livre (je n'en attendais pas moins de la part des auteurs ) mais surtout par la pédagogie employée pour expliquer des concepts qui peuvent, parfois, sembler compliquer/déroutant à des développeurs amateurs ou ne connaissant pas le sujet.

Tout ceux qui ont déjà écrit des livres (ou encore des articles) savent qu'il n'est pas simple d'expliquer et de retranscrire des notions complexes.

Les auteurs de ce livre ont parfaitement réussi ce challenge: il est donc à la fois très technique (n'oubliez pas que LINQ n'est pas une mince affaire) mais très bien expliqué, ce qui le rend d'autant plus intéressant !

Bref, pour tous ceux qui veulent en savoir plus sur les entrailles de LINQ, son fonctionnement, etc... ou bien pour tout ceux qui veulent savoir comment l'utiliser (et bien l'utiliser), alors je ne saurais que trop recommander ce livre !

## C# 3.0 In a Nutshell

### Critique du livre par Jérôme Lambert

Après de nombreuses lectures d'ouvrages sur C# 2.0 en anglais et en français, "C# 3.0 IN A NUTSHELL" fut mon premier livre sur la dernière version du langage C#, c'est-à-dire C# 3.0. Comme indiqué sur la couverture du livre, c'est la 3e édition avec comme nouveaux sujets couverts LINQ et les nouvelles classes du .NET Framework 3.5.

Autant le dire clairement, pour ceux qui connaissent C# 2.0, vous vous concentrerez tout comme moi sur les trois chapitres couvrant le nouveau projet du Framework .NET : LINQ. Et vous n'allez pas être déçu car ce n'est pas une petite mise à jour qui vous est proposée mais une centaine de pages sur :

- Les requêtes LINQ
- Les opérateurs LINQ
- L'Utilisation de LINQ avec XML

La première partie commencera par vous donner les bases de LINQ pour vous plonger assez vite dans une utilisation avancée avec l'exécution différée, les sous requêtes, les requêtes interprétées, LINQ to SQL et la construction des expressions d'une requête LINQ. Outre les nombreux exemples, les auteurs ont mis en avant la compréhension du mécanisme de LINQ auprès du lecteur. Vous apprendrez donc deux choses : l'utilisation de LINQ et la compréhension du mécanisme sous-jacent.

Pour la seconde partie concernant les opérateurs de LINQ, vous allez tout simplement avoir une vue d'ensemble des opérateurs de requête disponible avec C# 3.0, c'est-à-dire les filtres, jointures, tries, groupement, etc.

Enfin la troisième partie porte évidemment sur XML avec LINQ avec l'arrivée du nouveau namespace System.Xml.Linq dans le .NET Framework 3.5. Vous découvrirez les classes et les APIs permettant de lire et écrire du XML, manipuler des schémas et feuilles de style, ainsi que l'utilisation de XPath et X-DOM.

Pour les autres ne connaissant pas très bien le Framework .NET et C#, vous pourrez trouver votre bonheur avec tout ce qu'il faut savoir sur C# en commençant par les bases du langage, l'utilisation des différents types jusqu'à une utilisation avancée du langage avec les délégués, les événements, les nouveautés du langage C# 3.0, les attributs, etc. Ensuite, vous verrez les fondamentaux du Framework avec les chaînes de caractères, les dates, le formatage et le parsing, mais surtout les collections. La suite, je ne vais pas la répéter car ce sont les chapitres que vous retrouverez plus bas dans la partie "Sommaire" qui aborderont chaque partie majeure du Framework à connaître. Cela va de comment utiliser les streams avec C# jusqu'à l'utilisation d'expressions régulières.

En résumé, ce livre est vraiment clair et bien détaillé même si je le conseille plutôt à un public ayant quand même une expérience avec la programmation en général.

Retrouvez ces critiques de livre sur la page livres .NET : [Lien30](#)



## Les derniers tutoriels et articles

### La compilation séparée

La compilation séparée désigne le fait de compiler plusieurs fichiers sources séparément puis de les lier ensuite pour générer le produit final qui peut être un exécutable par exemple. Elle comprend plusieurs techniques que nous allons explorer tout au long de ce tutoriel.

#### 1. Généralités

##### 1.1. Compilation d'un projet

###### 1.1.1. Introduction

Un projet d'application en langage C est au moins constitué d'un fichier source qui sera compilé, ce qui donnera un fichier objet en sortie, puis lié à d'autres fichiers objets pour générer la sortie finale qui peut être un exécutable par exemple. Dans le cas général, un projet est constitué de plusieurs fichiers sources.

###### 1.1.2. Exemple avec un projet constitué de deux fichiers sources

###### 1.1.2.1. Le projet

Nous allons créer un programme qui affiche la somme et la différence de deux nombres entiers en séparant le programme et les fonctions (**somme** et **produit**) dans deux fichiers différents.

###### Fichier : exemple.c

```
#include <stdio.h>

int somme(int a, int b);
int produit(int a, int b);

int main(){
    int a = 2, b = 5;

    printf("%d + %d = %d\n", a, b, somme(a, b));
    printf("%d * %d = %d\n", a, b, produit(a, b));

    return 0;
}
```

###### Fichier : somme.c

```
int somme(int a, int b){
    return a + b;
}

int produit(int a, int b){
    int prod = 0;

    while (b-- > 0)
        prod += a;

    return prod;
}
```

###### 1.1.2.2. Compilation sous Code::Blocks

Pour compiler le projet, sélectionnez la commande **Build > Build (Ctrl + F9)**. Cette compilation se fait en deux phases : **compilation des fichiers sources** (exemple.c et fonctions.c) puis **édition des liens**. On peut aussi compiler les fichiers sources individuellement : **Build > Compile Current File (Ctrl + Shift + F9)**. Dans ce cas la commande **Build (Ctrl + F9)** ne fera plus que l'édition des liens.

#### 1.2. Le mot-clé extern

En langage C, tout objet (variable ou fonction) doit toujours avoir été déclaré avant d'être utilisé. Nous avons déjà résolu le problème pour les fonctions, ne reste donc plus que les variables. Supposons donc que l'on souhaite, depuis un fichier donné, accéder à une variable globale définie dans un autre fichier. On ne peut pas tout simplement déclarer une deuxième fois la variable car on aurait alors deux variables de même nom au sein d'un même projet ce qui conduirait à une erreur lors de l'édition des liens. Le mot-clé **extern** permet de résoudre le problème. Placé devant une déclaration, il permet d'indiquer que la variable ou fonction est définie (plus précisément : peut être définie) dans un autre fichier (source ou compilé). Evidemment si l'objet déclaré est une fonction, ce mot-clé ne sert qu'à la déco !

#### 1.3. Le mot-clé static

Devant la déclaration d'une variable globale ou d'une fonction, le mot-clé **static** restreint la visibilité de la variable ou de la fonction au fichier source courant. On a alors ce qu'on appelle une variable ou fonction privée. Uniquement pour les fonctions : si la fonction est définie avant sa première utilisation auquel cas elle ne nécessite donc pas de déclaration, alors on met tout simplement le mot-clé **static** devant la définition. En effet, il est syntaxiquement correct de placer ce mot-clé devant la déclaration ou la définition d'une fonction.

#### 1.4. Les fichiers d'en-tête

Les **fichiers d'en-tête** (\*.h) permettent de rassembler des « en-têtes » (c'est-à-dire des déclarations de fonctions, des définitions de macros, de types, etc.) communes à plusieurs fichiers sources et/ou fichiers d'en-tête. Mais puisqu'ils peuvent justement être inclus par un grand nombre de fichiers, le risque d'être inclus plus d'une fois dans un même fichier est très élevé. C'est pour cette raison que les fichiers d'en-têtes doivent impérativement être protégés contre les inclusions multiples. La technique fait appel au préprocesseur : une macro indique si le fichier est déjà inclus. Il suffit donc de tester dès le début du fichier si la macro est définie ou non. Le schéma est donc le suivant :

```
#ifndef DRAPEAU
```

```
#define DRAPEAU
```

```
/* ----- */  
/*      */  
/* ----- */
```

```
#endif
```

Dans l'exemple de projet précédent, on aurait pu par exemple créer un fichier **somme.h** contenant la déclaration des fonctions **somme** et **produit** du fichier **somme.c**.

```
Fichier : somme.h
```

```
#ifndef H_SOMME
```

```
#define H_SOMME
```

```
int somme(int a, int b);  
int produit(int a, int b);
```

```
#endif
```

Le fait d'avoir choisi `H_SOMME` plutôt que `SOMME_H` ou `__SOMME_H__` comme drapeau n'est pas du tout le fruit du hasard. Il permet de ne pas entrer en conflit avec les identifiants réservés du langage. Par exemple, les identifiants commençant par `E`, `LC_`, `SIG`, etc. sont réservés (`E` pour les numéros d'erreur de `errno.h`, `LC_` pour les constantes définies par `locale.h` et `SIG` pour les signaux de `signal.h`). `H_` en début d'un identifiant est pour l'heure encore libre, alors en profiter.

Il suffit maintenant d'inclure ce fichier partout où on a besoin des fonctions `somme` et `produit`. Par exemple :

```
Fichier : exemple.c
```

```
#include <stdio.h>  
#include "somme.h"
```

```
int main(){  
    int a = 2, b = 5;  
  
    printf("%d + %d = %d\n", a, b, somme(a, b));  
    printf("%d * %d = %d\n", a, b, produit(a, b));  
  
    return 0;  
}
```

Lorsque le nom d'un fichier est mis entre guillemets comme dans cet exemple, le préprocesseur va d'abord chercher le fichier dans le même répertoire que celui du fichier source puis s'il ne le trouve pas, va le chercher dans le ou les répertoires par défaut (spécifiques du compilateur). On peut également spécifier un chemin absolu.

## 1.5. Les structures opaques

Une structure est dite **opaque** lorsque son implémentation est cachée. L'accès aux champs de la structure se fait alors par l'intermédiaire de fonctions dont l'implémentation évidemment est également cachée. En particulier, l'interface devra au moins fournir une fonction permettant de créer l'objet (**constructeur**) et une fonction permettant de le détruire (**destructeur**). Par exemple, nous allons encapsuler le type `int` dans une structure de type `integer_s`.

Implémentons la structure dans un fichier **integer.c**

```
Fichier : integer.c
```

```
#include <stdlib.h>
```

```
struct integer_s {  
    int value;  
};
```

```
struct integer_s * integer_create_object(){  
    return malloc(sizeof(struct integer_s));  
}
```

```
void integer_set_value(struct integer_s * p_object, int value){  
    p_object->value = value;  
}
```

```
int integer_get_value(struct integer_s * p_object){  
    return p_object->value;  
}
```

```
void integer_delete_object(struct integer_s * p_object)  
{  
    free(p_object);  
}
```

Fournissons maintenant l'interface via un fichier d'en-tête :

```
integer.h
```

```
#ifndef H_INTEGER
```

```
#define H_INTEGER
```

```
struct integer_s;
```

```
struct integer_s * integer_create_object(void);  
void integer_set_value(struct integer_s * p_object, int value);  
int integer_get_value(struct integer_s * p_object);  
void integer_delete_object(struct integer_s * p_object);
```

```
#endif
```

La ligne :

```
struct integer_s;
```

**déclare** la structure (sans la définir). C'est ce qu'on appelle une **déclaration incomplète**. La structure est alors **opaque**.

Voici un exemple d'utilisation de `integer.h` :

```
exemple.c
```

```
#include <stdio.h>  
#include "integer.h"
```

```
int main(){  
    struct integer_s * i = integer_create_object();
```

```
    if (i != NULL){  
        integer_set_value(i, 100);  
        printf("The value of i is : %d\n",  
integer_get_value(i));  
        integer_delete_object(i);  
    }
```

```
    return 0;  
}
```

## 2. Les bibliothèques

### 2.1. Introduction

Une **bibliothèque** (**library** en anglais) est en première approximation un bouquet de fonctions. Chez certains langages elles sont appelées unités ou paquetages mais le principe reste le même. En langage C, il faut en fait également fournir le ou les fichiers d'en-tête correspondants (contenant la déclaration des fonctions de la bibliothèque, des macros et/ou types supplémentaires, etc.) avant de réellement en constituer une.

La manière de créer et d'utiliser une bibliothèque est très dépendante de l'environnement avec lequel on travaille. Dans ce tutoriel nous allons expliquer essentiellement la procédure pour **MS Visual Studio .NET** donc évidemment sous Windows.

Comme nous le savons déjà, la compilation d'un fichier source ne produit pas un exécutable mais un **module objet** (**.o** ou **.obj**), que l'on peut considérer comme la version machine du fichier source original. Il faut ensuite lier différents modules objets pour produire un exécutable.

Un module objet est réutilisable (on peut donc voir un tel fichier comme une véritable « **boîte à outils** »). C'est là d'ailleurs toute l'importance de la compilation séparée. Reprenons par exemple le fichier **somme.c** contenant le code des fonctions **somme** et **produit**. En compilant ce fichier, on obtient un fichier **somme.obj**.

Maintenant, créons un **nouveau projet** dans lequel nous allons utiliser les fonctions somme et produit. Voici le programme :

Fichier : exemple.c

```
#include <stdio.h>

int somme(int a, int b);
int produit(int a, int b);

int main(){
    int a = 2, b = 5;

    printf("%d + %d = %d\n", a, b, somme(a, b));
    printf("%d * %d = %d\n", a, b, produit(a, b));

    return 0;
}
```

Si on compile ce fichier, il n'y a aucune erreur puisque tout est syntaxiquement correct, l'équivalent en langage machine peut donc être généré. Par contre si on tente de générer l'exécutable, on aura un message d'erreur indiquant que l'édition des liens a échoué car les fonctions somme et produit n'ont pu être trouvés. Il faut donc dire au linkeur qu'il doit également chercher dans somme.obj lors de l'édition des liens. La procédure est évidemment dépendante de l'environnement de développement. Sous **Code::Blocks**, c'est dans **Project > Build Options > Linker > Link Libraries**. Sous **Visual Studio .NET** c'est **Project > Properties > Configuration Properties > Linker > Input > Additional Dependencies**. Il suffit ensuite d'ajouter **somme.obj**. Ce n'est pas plus différent non plus avec les autres EDIs.

Evidemment si vous ne spécifiez pas de chemin complet, le linqueur va supposer que le fichier se trouve dans le répertoire par défaut pour les libs (généralement un dossier nommé LIB dans le répertoire d'installation du compilateur) qui est bien entendu spécifique du linqueur.

### 2.2. Les bibliothèques statiques

Une **bibliothèque statique** (**.lib** ou **.a**) est un fichier qui regroupe un ou plusieurs modules objets. Elles s'utilisent donc de la même manière que ces derniers. Pour créer une bibliothèque statique avec **Visual Studio .NET**, créez un **nouveau projet Win32** (Win32 Project) puis dans **Paramètres de l'application** (Application settings), choisissez **Bibliothèque statique** (Static library). Cochez l'option **Projet vide** (Empty project) afin qu'aucun fichier source ne soit automatiquement ajouté au projet. A la fin, compilez le projet à l'aide du menu **Générer** (Build).

### 2.3. Les bibliothèques dynamiques

Une **bibliothèque dynamique** est un fichier qui ne sera effectivement lié à l'exécutable que pendant l'exécution. Cela présente plusieurs avantages. Supposez par exemple que vous avez créé une bibliothèque statique et que vous l'avez ensuite utilisé dans de nombreuses applications. Si un jour vous la modifiez et que vous voulez également mettre à jour toutes vos applications, vous devrez les recompiler une par une ! Pourtant si vous avez utilisé une bibliothèque dynamique, la modification seule de ce fichier aura des répercussions sur toutes les applications l'utilisant puisque la liaison avec le fichier ne se fait que pendant l'exécution. De plus, si vous avez bien compris, l'utilisation des bibliothèques dynamiques rend les exécutables plus petits (en terme de taille) puisque ce dernier même est incomplet. En effet, il a besoin du code contenu dans la bibliothèque pour fonctionner.

Sous Windows, les bibliothèques dynamiques sont appelées **DLLs** (**.dll**) pour Dynamic-Link Library. Sous UNIX on les appelle **Shared Objects** (**.so**). Dans ce tutoriel, nous-nous intéresserons aux DLLs.

Nous allons donc créer une DLL (**dsomme.dll**) exportant deux fonctions : **somme** et **produit**. Que signifie exporter ? Ben c'est très simple : lorsqu'on développe une bibliothèque, on peut spécifier quelles fonctions (ou variables) seront « **publiques** » (ou **exportées**), c'est-à-dire accessibles depuis l'extérieur, et lesquelles seront « **privées** », c'est-à-dire réservées à usage interne. Nous avons déjà vu que le mot-clé **static** permet de rendre une fonction privée. Cependant dans le cas d'une DLL, toutes les fonctions sont par défaut privées ! Le mot-clé **static** ne nous sert donc plus à grandchose.

La question est donc maintenant : comment exporter une fonction. Et ben il y a plusieurs manières de le faire, par exemple à l'aide du modificateur **\_\_declspec(dllexport)**. Bien entendu, il s'agit bien d'une extension Microsoft aux langages C et C++ (en ce qui nous concerne : le langage C) et qui fut ensuite repris par la plupart des implémentations pour Windows. Donc pas de problème que vous compilez avec MingW ou Borland C++ ...

Sous Visual Studio .NET, créez un **nouveau projet Win32** (Win32 Project) puis dans **Paramètres de l'application** choisissez **DLL**. Cochez l'option **Projet vide** afin qu'aucun fichier ne soit automatiquement ajouté au projet. Ajoutez ensuite un fichier **dsomme.c** puis saisissez le code suivant :

Fichier : dsomme.c

```
__declspec(dllexport) int somme(int a, int b){
    return a + b;
}

__declspec(dllexport) int produit(int a, int b){
    int prod = 0;
}
```

```
while (b-- > 0)
    prod += a;

return prod;
}
```

Compilez ensuite le projet avec la commande **Build** du menu Build. Vous obtiendrez entre autres en sortie deux fichiers : **dsomme.dll** et **dsomme.lib**. Ce dernier, bien que portant l'extension .lib, n'est pas une bibliothèque statique mais une **bibliothèque d'importation**. C'est lui qu'il faut passer au linker lors de l'édition des liens pour pouvoir compiler du code dépendant d'une DLL. A l'exécution, le programme doit pouvoir localiser la DLL. Cette dernière doit donc se trouver soit dans le même répertoire que le programme, soit dans le répertoire courant du programme, ou encore dans un répertoire « connu » du système par exemple le répertoire system32.

La déclaration de fonctions à importer depuis une DLL (via la bibliothèque d'importation) peut se faire comme la déclaration d'une fonction « normale », cependant le modificateur **\_\_declspec(dllexport)** permet d'indiquer au compilateur (je dis bien le compilateur, pas le linker) que la fonction en question se trouve dans une bibliothèque dynamique, ce qui lui permettra de générer du code plus efficace (plus « direct »). Sans cela, le compilateur va tout simplement convertir chaque appel de la fonction en appel de celle qui se trouve dans le .lib, qui ne fait rien de plus qu'un appel à la fonction dans la DLL ce qui fait donc finalement deux appels, ce qui est évidemment plus long qu'un appel direct. Notre programme sera donc :

#### Fichier : exemple.c

```
#include <stdio.h>

__declspec(dllexport) int somme(int a, int b);
__declspec(dllexport) int produit(int a, int b);

int main(){
```

```
int a = 2, b = 5;

printf("%d + %d = %d\n", a, b, somme(a, b));
printf("%d * %d = %d\n", a, b, produit(a, b));

return 0;
}
```

Et n'oubliez pas : nous devons nous lier avec dsomme.lib.

## 2.4. Applications. Exemples

### 2.4.1. La bibliothèque standard du langage C

Sous Windows, la bibliothèque « standard » du langage C, c'est-à-dire celle qui contient entre autres le code des fonctions standard du C est implémentée en tant que bibliothèque dynamique connue sous le nom de Microsoft C Run-Time Library (CRT), et qui correspond au fichier MSVCRT.DLL. On peut toujours bien sûr se lier statiquement avec cette bibliothèque (il suffit de faire les réglages nécessaires, ça dépend du compilateur) mais dans ce cas les exécutables seront considérablement plus gros.

### 2.4.2. Le concept d'API

Une API ou Application Programming Interface (Interface de Programmation d'Applications) est un ensemble de fonctions exposées par un système ou un logiciel pour permettre à d'autres logiciels d'interagir (c'est-à-dire de communiquer) avec lui. Par extension, toute fonction d'une API donnée est également appelée : une API. Sous UNIX, les APIs du système sont appelés appels système.

Les DLLs sont très utilisées sous Windows et le système lui-même expose son API via de nombreuses DLLs. Les programmes conçus spécifiquement pour Windows se lient donc à un ou plusieurs DLLs de l'API Windows.

Retrouvez l'article de Jesse Edouard en ligne : [Lien103](#)

## Développez et déployez une application GTK+ sous Windows

Cet article va vous expliquer comment développer puis déployer une application écrite grâce à la bibliothèque GTK+ sous Windows.

### 1. Introduction

Sous Linux pour développer une application à l'aide de GTK+, c'est relativement simple. Pour cela, il suffit de s'appuyer sur les outils fournis par votre distribution : apt-get ou yum, par exemple, pour l'installation des bibliothèques, les autotools pour la compilation et quelques dizaines d'éditeurs de texte, IDE ou autre RAD pour le développement. Il en va de même pour la distribution de votre application.

L'un des (nombreux) avantages de GTK+ est la possibilité de disposer du même outil sous Linux et sous Windows. Cependant les utilisateurs et développeurs disposant de ce dernier ne sont pas les mieux servis : aucun système pour gérer les dépendances, une ligne de commande pauvre, ...

Dans ce tutoriel je vais donc vous présenter comment installer le nécessaire pour développer une application à l'aide de GTK+ et ensuite comment créer un installateur pour distribuer facilement votre création.

### 2. Installation

#### 2.1. Code::Blocks

Pour vos développements utilisant le C ou le C++ sous Windows, je vous conseille l'IDE Code::Blocks ([Lien104](#)). Son installation est extrêmement simple, téléchargez et installez les paquets suivant :

- MinGW-5.1.3.exe : [Lien105](#)
- gdb-6.6.tar.bz2 : [Lien106](#)
- Code::Blocks 8.02 : [Lien107](#)

Installez mingw (gcc, g++ et make), à la racine du disque C: (C:\MinGW) et décompressez gdb au même endroit. Ensuite installez Code::Blocks dans le répertoire des programmes (C:\Programs Files\CodeBlocks). Voilà vous pouvez dès à présent utiliser Code::Blocks.

Passons maintenant à l'installation des bibliothèques GTK+.

## 2.2. GTK+

L'installation est encore plus simple. Allez sur le site officiel de GTK+ ([Lien108](#)) et téléchargez tous les packages binaires et dev sans oublier les dépendances requises :

- GLib
- GTK+
- Pango
- ATK
- Cairo
- zlib
- gettext-runtime
- libpng
- libjpeg
- libtiff

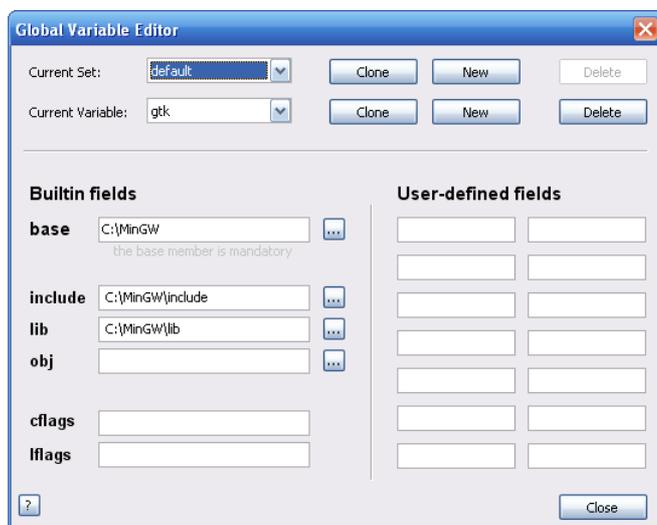
Et dézipper tout à la racine du répertoire de mingw !

**Attention** : Bizarrement le fichier zlib.dll se retrouve à la racine, pour éviter tout problème, déplacez le fichier dans le sous-répertoire bin.

## 2.3. Test

Maintenant il nous reste plus qu'à tester tout ça ! Commencez par lancer Code::Blocks. Vérifiez dans Settings -> Compiler and debugger onglet Toolchain executables que le répertoire correspond bien à celui choisi (par défaut C:\MinGW).

Ensuite Settings -> Global Variable... pour créer une variable qui permettra à Code::Blocks de trouver nos fichiers d'entête et nos bibliothèques lors de la création d'un nouveau projet. Voici la configuration par défaut :



Pour finir File -> New -> Project..., choisissez le template GTK+ project. Lorsque l'assistant de configuration vous demande l'emplacement de GTK+, entrez simplement \${#gtk}.

Pressez la touche F9 et admirez le résultat !

## 3. Déploiement

Maintenant votre application compile et s'exécute correctement, mais uniquement sous Code::Blocks. Essayez d'exécuter directement votre programme (dans le sous-répertoire bin\Debug de votre projet) et vous devriez obtenir un beau message d'erreur :



Tout simplement parce que vos dll ne se trouvent pas dans l'un des répertoires présent dans la variable d'environnement PATH. Vous pouvez bien sûr l'ajouter mais qu'en sera-t-il lorsque vous voudrez distribuer votre programme ? Vous souhaitez faire subir la même punition aux utilisateurs de vos programmes ? Pas très motivant pour utiliser votre programme. Il existe bien sûr la solution de lier les bibliothèques statiquement à votre programme, mais au cours de mes tests je n'ai jamais réussi... De plus cela vous oblige à publier votre projet sous licence libre.

Nous allons donc partir sur une solution plus propre et plus professionnelle basée sur un exécutable auto-extractable créé grâce à inno setup.

Une fois installé, lancez le programme. Vous obtenez un "simple" éditeur de fichier texte. Voici un fichier type qui contient l'ensemble des fichiers dont un programme utilisant GTK+ a besoin pour s'exécuter.

N'oubliez pas de créer un fichier COPYING.txt contenant la licence de votre programme.

```
; Script generated by the Inno Setup Script Wizard.
; SEE THE DOCUMENTATION FOR DETAILS ON CREATING INNO
SETUP SCRIPT FILES!
```

```
[Setup]
AppName=test
AppVerName=test 0.1
AppPublisher=developpez.com
AppPublisherURL=http://www.developpez.com/
AppSupportURL=http://www.developpez.com/
AppUpdatesURL=http://www.developpez.com/
DefaultDirName={pf}\test
DefaultGroupName=test
AllowNoIcons=yes
LicenseFile=COPYING.txt
OutputBaseFilename=setup
Compression=lzma
SolidCompression=yes

[Languages]
Name: "french"; MessagesFile:
"compiler:Languages\French.isl"

[Tasks]
Name: "desktopicon"; Description:
"{cm:CreateDesktopIcon}"; GroupDescription:
"{cm:AdditionalIcons}"; Flags: unchecked
Name: "quicklaunchicon"; Description:
"{cm:CreateQuickLaunchIcon}"; GroupDescription:
"{cm:AdditionalIcons}"; Flags: unchecked
```

```
[Files]
Source: "C:\Documents and Settings\gege2061\Mes
documents\test\bin\Release\test.exe"; DestDir: "{app}";
Flags: ignoreversion

; GTK+ dependencies
; DLL
Source: "C:\MinGW\bin\libcairo-2.dll"; DestDir:
"{app}"; Flags: ignoreversion
Source: "C:\MinGW\bin\libpangocairo-1.0-0.dll";
DestDir: "{app}"; Flags: ignoreversion
Source: "C:\MinGW\bin\jpeg62.dll"; DestDir: "{app}";
Flags: ignoreversion
Source: "C:\MinGW\bin\libtiff3.dll"; DestDir: "{app}";
Flags: ignoreversion
Source: "C:\MinGW\bin\libpng13.dll"; DestDir: "{app}";
Flags: ignoreversion
Source: "C:\MinGW\bin\zlib1.dll"; DestDir: "{app}";
Flags: ignoreversion
```

```
Source: "C:\MinGW\bin\intl.dll"; DestDir: "{app}";
Flags: ignoreversion
Source: "C:\MinGW\bin\libatk-1.0-0.dll"; DestDir:
"{app}"; Flags: ignoreversion
Source: "C:\MinGW\bin\libgdk_pixbuf-2.0-0.dll";
DestDir: "{app}"; Flags: ignoreversion
Source: "C:\MinGW\bin\libgdk-win32-2.0-0.dll"; DestDir:
"{app}"; Flags: ignoreversion
Source: "C:\MinGW\bin\libglib-2.0-0.dll"; DestDir:
"{app}"; Flags: ignoreversion
Source: "C:\MinGW\bin\libgmodule-2.0-0.dll"; DestDir:
"{app}"; Flags: ignoreversion
Source: "C:\MinGW\bin\libgobject-2.0-0.dll"; DestDir:
"{app}"; Flags: ignoreversion
Source: "C:\MinGW\bin\libgthread-2.0-0.dll"; DestDir:
"{app}"; Flags: ignoreversion
Source: "C:\MinGW\bin\libgtk-win32-2.0-0.dll"; DestDir:
"{app}"; Flags: ignoreversion
Source: "C:\MinGW\bin\libpango-1.0-0.dll"; DestDir:
"{app}"; Flags: ignoreversion
Source: "C:\MinGW\bin\libpangoft2-1.0-0.dll"; DestDir:
"{app}"; Flags: ignoreversion
Source: "C:\MinGW\bin\libpangowin32-1.0-0.dll";
DestDir: "{app}"; Flags: ignoreversion
```

; .mo

```
Source: "C:\MinGW\lib\locale\fr\LC_MESSAGES\atk10.mo";
DestDir: "{app}\lib\locale\fr\LC_MESSAGES"; Flags:
ignoreversion
Source:
"C:\MinGW\share\locale\fr\LC_MESSAGES\glib20.mo";
DestDir: "{app}\share\locale\fr\LC_MESSAGES"; Flags:
ignoreversion
Source:
"C:\MinGW\share\locale\fr\LC_MESSAGES\gtk20.mo";
DestDir: "{app}\share\locale\fr\LC_MESSAGES"; Flags:
ignoreversion
Source: "C:\MinGW\share\locale\fr\LC_MESSAGES\gtk20-
properties.mo"; DestDir:
"{app}\share\locale\fr\LC_MESSAGES"; Flags:
ignoreversion
```

;

```
Source: "C:\MinGW\etc\gtk-2.0\gdk-pixbuf.loaders";
DestDir: "{app}\etc\gtk-2.0"; Flags: ignoreversion
Source: "C:\MinGW\etc\gtk-2.0\gtk.immodules"; DestDir:
"{app}\etc\gtk-2.0"; Flags: ignoreversion
Source: "C:\MinGW\etc\pango\pango.modules"; Destdir:
"{app}\etc\pango"
Source: "C:\MinGW\etc\pango\pango.aliases"; Destdir:
"{app}\etc\pango"
```

; optional: let the user make the app look more  
Windows-like

```
Source:
"C:\MinGW\lib\gtk-2.0\2.10.0\engines\libwimp.dll";
Destdir: "{app}\lib\gtk-2.0\2.10.0\engines"
Source:
"C:\MinGW\lib\gtk-2.0\2.10.0\engines\libpixmap.dll";
Destdir: "{app}\lib\gtk-2.0\2.10.0\engines"
Source: "C:\MinGW\lib\gtk-2.0\2.10.0\immodules\im-am-
et.dll"; Destdir: "{app}\lib\gtk-2.0\2.10.0\immodules"
Source: "C:\MinGW\lib\gtk-2.0\2.10.0\immodules\im-
cedilla.dll"; Destdir:
"{app}\lib\gtk-2.0\2.10.0\immodules"
Source: "C:\MinGW\lib\gtk-2.0\2.10.0\immodules\im-
cyrillic-translit.dll"; Destdir:
"{app}\lib\gtk-2.0\2.10.0\immodules"
Source: "C:\MinGW\lib\gtk-2.0\2.10.0\immodules\im-
ime.dll"; Destdir: "{app}\lib\gtk-2.0\2.10.0\immodules"
Source: "C:\MinGW\lib\gtk-2.0\2.10.0\immodules\im-
inuktitut.dll"; Destdir:
"{app}\lib\gtk-2.0\2.10.0\immodules"
Source: "C:\MinGW\lib\gtk-2.0\2.10.0\immodules\im-
ipa.dll"; Destdir: "{app}\lib\gtk-2.0\2.10.0\immodules"
Source: "C:\MinGW\lib\gtk-2.0\2.10.0\immodules\im-
multipress.dll"; Destdir:
"{app}\lib\gtk-2.0\2.10.0\immodules"
Source: "C:\MinGW\lib\gtk-2.0\2.10.0\immodules\im-
thai.dll"; Destdir:
"{app}\lib\gtk-2.0\2.10.0\immodules"
Source: "C:\MinGW\lib\gtk-2.0\2.10.0\immodules\im-ti-
er.dll"; Destdir: "{app}\lib\gtk-2.0\2.10.0\immodules"
Source: "C:\MinGW\lib\gtk-2.0\2.10.0\immodules\im-ti-
et.dll"; Destdir: "{app}\lib\gtk-2.0\2.10.0\immodules"
Source: "C:\MinGW\lib\gtk-2.0\2.10.0\immodules\im-
vigr.dll"; Destdir:
"{app}\lib\gtk-2.0\2.10.0\immodules"
```

[Run]

```
Filename: "{app}\test.exe"; Description:
"{cm:LaunchProgram,test}"; Flags: nowait postinstall
skipifsilent
```

Pour obtenir votre installateur, tapez ctrl+F9 et vous obtenez un exécutable setup.exe dans le sous-répertoire Output. Exécutez-le et une fois terminé lancez de nouveau votre programme... Admirez le résultat !

Retrouvez la suite de l'article de Nicolas Joseph en ligne : [Lien108](#)

## Les derniers tutoriels et articles

### Firestarter : le pare-feu en toute simplicité

#### 1. Présentation et Installation

Firestarter est une interface graphique qui vous aide à configurer facilement votre pare-feu. Il a pour objectif d'être le plus simple possible tout en restant complet et efficace.

Utilisez le gestionnaire de packages de votre distribution ou consultez le site officiel ([Lien109](#)).

#### 2. Premier lancement

Pour lancer Firestarter, cliquez sur Système > Administration > Firestarter. Au premier lancement, un assistant vous aide à configurer Firestarter en choisissant l'interface à surveiller (il a normalement détecté automatiquement l'interface active). Si votre adresse IP vous est attribuée automatiquement via DHCP, cochez l'option correspondante.

La seconde boîte de dialogue vous propose de partager votre connexion et d'utiliser votre machine comme serveur DHCP. Si vous avez votre PC directement relié à Internet et que vous partagez avec d'autres ordinateurs "derrière vous", cochez les 2 cases, sinon cliquez sur Avancer puis Enregistrer pour terminer l'assistant.

Si vous installez une nouvelle interface réseau par la suite, vous pourrez (et même vous devriez si le pare-feu bloque votre connexion) relancer l'assistant en allant dans le menu Pare-feu > Lancer l'assistant. Pour les utilisateurs avertis, il est aussi possible de passer par le menu Édition > Préférences.

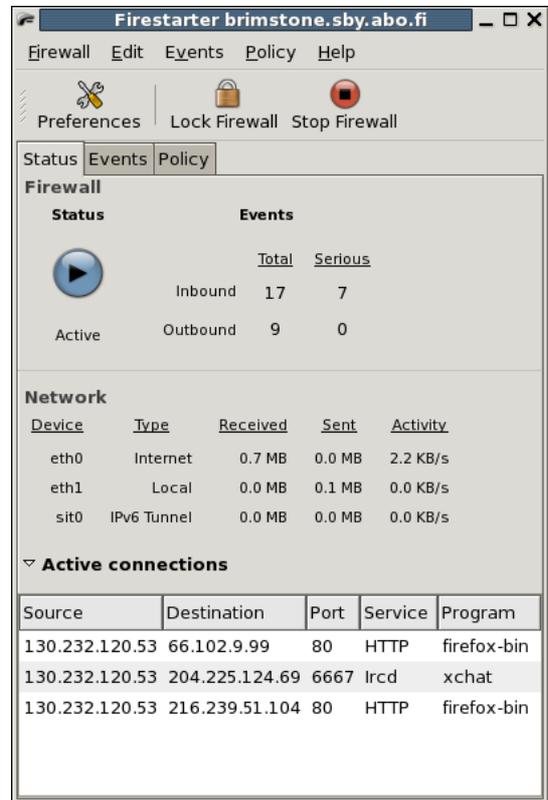
Si vous ne faites pas tourner de serveur sur votre machine, que les logs ne vous intéressent pas et que vous ne souhaitez pas vous plonger davantage dans la configuration de cet outil, vous pouvez vous arrêter ici et en rester à la configuration par défaut, qui devrait vous satisfaire.

#### 2.1. L'onglet « État »

Cet onglet montre et permet de contrôler l'état général du pare-feu, qui peut être :

- Actif : le pare-feu est en train de faire son travail ;
- Arrêté : le pare-feu est désactivé, il n'agit en rien sur le trafic ;
- Bloqué : le pare-feu bloque complètement le trafic entrant et sortant, ie rien ne passe.

Cette page propose également quelques statistiques sur le trafic internet telles que les connexion actives, le nombres d'alertes et de paquets reçu, etc.



The screenshot shows the Firestarter interface for 'brimstone.sby.abo.fi'. It has a menu bar with 'Firewall', 'Edit', 'Events', 'Policy', and 'Help'. Below the menu are buttons for 'Preferences', 'Lock Firewall', and 'Stop Firewall'. The main area is divided into 'Status', 'Events', and 'Policy' tabs. The 'Status' tab shows the Firewall is 'Active' with a play button icon. It also displays a table of 'Events' with columns for 'Total' and 'Serious'. The 'Network' section shows a table of active connections with columns for 'Device', 'Type', 'Received', 'Sent', and 'Activity'. The 'Active connections' section shows a table with columns for 'Source', 'Destination', 'Port', 'Service', and 'Program'.

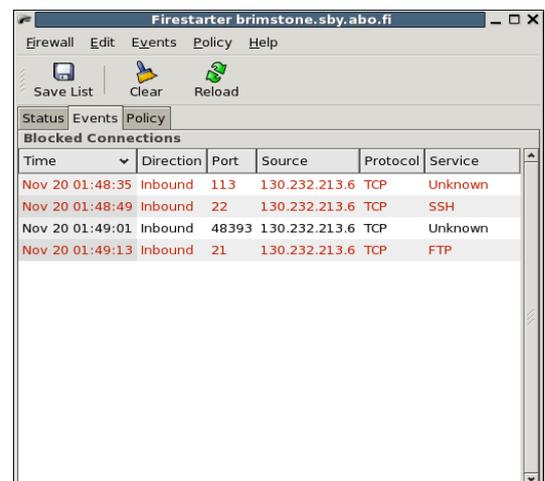
Device	Type	Received	Sent	Activity
eth0	Internet	0.7 MB	0.0 MB	2.2 KB/s
eth1	Local	0.0 MB	0.1 MB	0.0 KB/s
sit0	IPv6 Tunnel	0.0 MB	0.0 MB	0.0 KB/s

Source	Destination	Port	Service	Program
130.232.120.53	66.102.9.99	80	HTTP	firefox-bin
130.232.120.53	204.225.124.69	6667	lrcd	xchat
130.232.120.53	216.239.51.104	80	HTTP	firefox-bin

#### 2.2. L'Onglet « Évènements »

C'est le coin des logs, où vous pouvez voir les tentatives de connexion bloquées avec leur degré de gravité :

- Noir : tentative de connexion régulière sur un port, bloquée par le pare-feu, en règle générale pas de quoi fouetter un chat ;
- Rouge : possible tentative d'intrusion, également bloquée ;
- Gris : connexions que Firestarter juge « non-dangereuses », en règle générale du trafic « broadcast ».



The screenshot shows the 'Blocked Connections' section of the Firestarter interface. It has a menu bar with 'Firewall', 'Edit', 'Events', 'Policy', and 'Help'. Below the menu are buttons for 'Save List', 'Clear', and 'Reload'. The main area is divided into 'Status', 'Events', and 'Policy' tabs. The 'Blocked Connections' section shows a table with columns for 'Time', 'Direction', 'Port', 'Source', 'Protocol', and 'Service'.

Time	Direction	Port	Source	Protocol	Service
Nov 20 01:48:35	Inbound	113	130.232.213.6	TCP	Unknown
Nov 20 01:48:49	Inbound	22	130.232.213.6	TCP	SSH
Nov 20 01:49:01	Inbound	48393	130.232.213.6	TCP	Unknown
Nov 20 01:49:13	Inbound	21	130.232.213.6	TCP	FTP

### 2.3. L'Onglet « Politique »

Définissez ici vos règles pour le trafic entrant et sortant de votre machine. Pour le trafic sortant, je suggère l'option par défaut. Pour le trafic entrant, si vous avez des serveurs tournant sur votre machine, ouvrez les ports correspondant :

1. clic droit dans la zone « Autoriser le service » ;
2. clic droit dans la zone « Autoriser le service » ;
3. « Ajouter une règle » ;
4. sélectionnez dans la liste de nom le nom du service que vous faites tourner (par exemple « FTP » pour un serveur FTP) ou, s'il n'y est pas, entrez le nom du service et le numéro du port ;
5. laissez l'option par défaut (« Tout le monde ») dans le champ « Source », afin d'ouvrir ce port pour tout le monde ;
6. validez en appuyant sur « Ajouter ».



### 2.4. Les préférences (Édition > Préférences)

Les préférences par défaut conviennent à l'utilisateur lambda. Ceux d'entre vous qui connaissent déjà les protocoles réseau et le firewalling n'auront aucune difficulté à s'y retrouver, je renvoie les autres curieux aux documents facilement trouvables sur le sujet des protocoles réseau et du firewalling.

#### Le pare-feu est-il actif lorsque la fenêtre de Firestarter est fermée ?

La fenêtre ne sert qu'à la configuration. Le pare-feu est donc actif même quand elle est fermée, selon ce que vous avez défini dans les préférences « Pare-feu ». Par défaut, le pare-feu (re)démontre au lancement d'une connexion et à l'ouverture de l'interface de configuration et à l'attribution d'une nouvelle adresse via DHCP, ce qui vous assure la couverture de vos arrières.

## 3. Annexes

### 3.1. Exemple configuration

J'insère ici ma configuration car elle fut pour moi très longue à trouver !

```
sudo firestarter
```

#### Légende

- \* = coché
- / = décoché

Préférence :

- \* Liste libre
- Interface :
  - / Activer l'icone dans la barre des tâches
  - / Minimiser dans la barre des tâches sur fermeture de la fenêtre
- Événements :
  - \* Omettre les entrées redondantes
  - / Omettre les entrées quand la destination n'est pas le pare-feu
- Politique :
  - \* Appliquer les changements de politique immédiatement
  - Pare-feu :
    - \* Démarrer/Redémarrer le pare-feu au démarrage du programme
    - \* Démarrer/Redémarrer le pare-feu lors d'une connexion par modem
    - \* Démarrer/Redémarrer le pare-feu sur une nouvelle adresse DHCP
  - Configuration du réseau :
    - Périphérique réseau connecté à Internet :
      - Périphériques détectés : [Périphérique ...]
    - Périphérique connecté au réseau local :
      - Périphériques détectés : [Périphérique ...]
      - / Autoriser le partage de la connexion Internet -
  - > / Pas de partage de connexion ?
    - / Autoriser le DHCP pour le réseau local
  - Filtrage ICMP
    - \* Autoriser le filtrage ICMP
    - Autoriser le suivi des types de paquet ICMP
      - / Requête par écho (ping)
      - / Réponse par écho (pong)
      - / Marquage temporel
    - / MS Traceroute
    - / Traceroute
    - / Inaccessible
    - / Masquage d'adresse
    - / Redirection
    - / Extinction de la source
  - Filtrage ToS :
    - \* Autoriser le filtrage sur le Type de Service (ToS)
  - Mettre une priorité plus forte pour :
    - / Les stations de travail
    - / Les serveurs
    - / Le système X Windows
  - Régler les priorités pour maximiser :
    - / le débit
    - \* la sûreté de fonctionnement
    - / l'interactivité
  - Options avancées :
    - Méthode préférée de rejet des paquets
      - / Rejeter les paquets avec une erreur
      - \* Rejeter silencieusement
    - Trafic broadcast :
      - / Bloquer le trafic broadcast du réseau externe

```
/ Bloquer le trafic broadcast du réseau interne
- Validation du trafic :
/ Bloquer le trafic des adresses réservées sur les
interfaces publiques
```

## 2.2. Lancer au démarrage Firestarter

Testé sur t60p avec ubuntu 7.04

Firestarter est un bon logiciel mais il pose quelques problèmes pour le lancer au démarrage de la session. En faite, lorsqu'on se contente d'ajouter firestarter dans système > préférences > sessions, il affiche un message d'erreur en disant que la carte eth0 par exemple n'est pas disponible . Ceci est compréhensible car il se lance en même temps que la connexion au réseau est la connexion n'a pas le temps de s'établir mais voici la solution le « script ».

Premièrement il faut vous autorisez définitivement à lancer firestarter :

```
sudo visudo
```

Ajoutez à la fin :

```
username ALL= NOPASSWD: /usr/sbin/firestarter
```

ne pas oublier de remplacer username par votre pseudo.

Pour d'autres versions de GNU/Linux il est probable qu'il faut remplacer sbin par bin.

Maintenant créez un fichier vierge qu'on appellera .start\_firestarter

```
touch .start_firestarter
```

Ouvrez-le :

```
gedit .start_firestarter
```

et copiez ce qui vous intéresse...

Avec deux interfaces réseau eth0 (wifi) et eth1 (filaire)

```
#!/bin/bash
```

```
verif=$(ifconfig eth0 | grep Octets | cut -d: -f2 |
cut -d' ' -f1)
verif1=$(ifconfig eth1 | grep Octets | cut -d: -f2 |
cut -d' ' -f1)
i="0"
```

```
while [ "$verif" -lt 900 ] && [ "$verif1" -lt 900 ] &&
```

```
[ "$i" -lt 100001 ]; do
    verif=$(ifconfig eth0 | grep Octets | cut -d:
-f3 | cut -d' ' -f1)
    verif1=$(ifconfig eth1 | grep Octets | cut -d: -
f2 | cut -d' ' -f1)
    let $[ i=i+1 ]
done

if [ "$i" -lt 100000 ] ; then
{
    sudo firestarter --start-hidden
}
fi

exit 0
```

Avec une interface réseau eth0

```
#!/bin/bash
```

```
## récupère le nombre d'octet reçus sur eth0
verif=$(ifconfig eth0 | grep Octets | cut -d: -f2 |
cut -d' ' -f1)
```

```
## compteur
```

```
i="0"
```

```
## Tant que la carte réseau n'a pas reçus 900 octets ou
que le compteur n'est pas fini
while [ "$verif" -lt 900 ] && [ "$verif1" -lt 900 ]; do
    verif=$(ifconfig eth0 | grep Octets | cut -d:
-f3 | cut -d' ' -f1)
```

```
    let $[ i=i+1 ]
```

```
done
```

```
## si la boucle s'est finie avant la fin du compteur
alors firestarter est lancé dans la barre de tache
```

```
if [ "$i" -lt 100000 ] ; then
```

```
{
    sudo firestarter --start-hidden
}
```

```
fi
```

```
exit 0
```

Pour finir, allez dans Système > Préférences > Sessions. Cliquez sur Ajouter :

```
Nom : firestarter
```

```
commande: sh /home/<<username>>/start_firestarter
```

Retrouvez l'article paru sur GuruLinux en ligne : [Lien110](#)

## Les livres Linux

### Debian Etch GNU/Linux

Debian GNU/Linux, distribution Linux non commerciale extrêmement populaire, est réputée pour sa fiabilité et sa richesse. Soutenue par un impressionnant réseau de développeurs dans le monde, elle a pour principes l'engagement vis-à-vis de ses utilisateurs et la qualité. Ses technologies concernent un nombre toujours croissant d'administrateurs, notamment par le biais de la distribution dérivée Ubuntu.

Ce cahier de l'Admin consacré à Debian Etch perpétue le succès de sa première version : accessible à tous, il fournit un ensemble suffisant de connaissances pour qui souhaite devenir un administrateur Debian GNU/Linux efficace et indépendant. Il traite des outils et méthodes qu'un administrateur Linux compétent maîtrise, depuis l'installation et la mise à jour du système jusqu'à la création de paquets et la compilation d'un noyau Linux, en passant par la supervision, la sauvegarde et les migrations, sans oublier des techniques avancées telles que la mise en place de SELinux pour sécuriser des services,

l'automatisation des installations ou encore la virtualisation avec Xen.

### **Critique du livre par Olivier Van Hoof**

C'est avec plaisir que j'ai abordé cet ouvrage, la version précédente consacrée à Debian Sarge étant déjà très réussie mais améliorée sur certains points, j'étais curieux de voir le résultat dans cette nouvelle mouture. Rien que le nombre de pages a considérablement augmenté, je m'attendais donc à des changements conséquents et je ne fus pas déçu, que du contraire !

La structure de base du livre reste identique et logique, à savoir : présentation de Debian et étude de cas typique, installation et administration de base du système, configuration plus avancée pour serveurs et stations de travail, et extension du système en apprenant à faire ses propres packages. Outre les mises à jour nécessaires en fonction des nouvelles versions des outils Debian, ce livre va plus loin que le précédent sur quasiment tous les points.

Dès l'installation, le partitionnement en RAID et LVM est abordé avec un renvoi vers un chapitre ultérieur pour des explications plus détaillées sur ces 2 technologies et leur gestion au quotidien. Après la description du système de packages propre à Debian, un chapitre spécial est consacré à la recherche d'informations et donne quelques pistes pour parer aux interrogations les plus courantes; excellent point pour les débutants donc.

La configuration des différents éléments du système est plus complète, notamment sur ssh avec l'authentification par clés, mais certains sujets restent toujours malheureusement survolés comme la configuration d'un réseau sans fil... Par contre un sujet important comme la compilation du noyau, absente de la première version du livre, est ici bien expliqué, avec les particularités de Debian. Les auteurs expliquent aussi les différentes manières de procéder à des installations automatisées, la fabrication d'images ISO du système... Au niveau réseau on trouve aussi une présentation plus complète des principaux outils de contrôle et de diagnostic d'un serveur ou d'un parc de machines. Un chapitre entier est d'ailleurs consacré à la sécurité, en passant en revue l'élaboration d'un firewall iptables, les outils de détection d'intrusion, les bonnes pratiques à adopter, et comment réagir en cas de piratage, ainsi que le système selinux. Enfin, cerise sur le gâteau, un chapitre entier explique les notions fondamentales des systèmes linux : séquence de boot, arborescence des répertoires, commandes shell de base pour apprendre à manipuler les fichiers, les processus, charger et décharger les modules du noyau, etc.

Bien plus complet que le précédent, cet ouvrage est une véritable référence pour tout qui veut installer et utiliser Debian quelque soit son niveau.

Retrouvez cette critique de livre sur la page livres Linux : [Lien11](#)



## Les blogs Mac

### Saviez-vous que les Mac n'ont pas de BIOS ?

Toute personne (du moins celles et ceux qui fréquentent developpez.com) sait que la majorité des ordinateurs personnels vendus aujourd'hui possèdent encore et toujours un BIOS. Et que c'est ce même BIOS qui est responsable d'aller lire le MBR qui va lui charger l'OS en mémoire (je simplifie grandement. Ne m'en veuillez pas. Si vous désirez en savoir plus sur le BIOS et MBR, allez voir le tutoriel de Baptiste Wicht ([Lien112](#))).

Mais saviez-vous que les MacIntel n'ont pas de BIOS ? Et qu'ils ne "bootent" pas grâce au MBR ?

Lors de l'abandon des processeurs PowerPC pour les processeurs Intel, Apple n'a pas choisi d'utiliser le BIOS, mais d'utiliser l'EFI.

EFI est l'abréviation de Extensible Firmware Interface.

L'EFI est quelque chose qu'Intel voudrait faire adopter par les constructeurs. Mais pour que les constructeurs adoptent l'EFI, il faudrait que les OS supportent l'EFI. Et, jusqu'à preuve du contraire, l'OS qui a la plus grande part de marcher actuellement, c'est Windows.

Et, d'après le site de Microsoft, seules les versions de Windows Server 2003 et 2008 supportent l'EFI pour la plateforme Intel Itanium. Ce qui exclut donc les processeurs Intel Core 2 Duo, qui sont ceux que l'on retrouve sur la plupart des PC vendus aujourd'hui. Windows Server 2008 devrait également supporter les processeurs Core 2 Duo d'Intel. Mais on ne peut pas dire que ce sont là les versions de Windows que l'on rencontre sur la majorité des PC vendus aujourd'hui. A ce jour, ni Windows XP, ni Windows Vista, sorti l'année passée, ne supportent l'EFI. (Notez qu'on retrouve un support de EFI 2.0 dans le Service Pack 1 de Windows Vista sorti récemment) Ce qui explique grandement pourquoi les cartes mères vendues aujourd'hui, autres que celles équipées du processeur Itanium, sont toujours équipées d'un Bios et n'ont généralement pas d'EFI.

Sauf en ce qui concerne les Mac. Apple, du fait qu'il vend une solution intégrée, a pu, sans aucun problème, ni aucun remord, adopter l'EFI.

Un autre avantage d'utiliser l'EFI au lieu du Bios, tient dans sa première lettre, E, qui signifie Extensible.

Apple pouvait donc étendre les fonctionnalités de l'EFI pour pouvoir proposer sous MacIntel les mêmes fonctionnalités qu'elle proposait déjà sur les Mac PowerPC, comme, par exemple, le

Target Mode (la possibilité de transformer son Mac en disque dur FireWire lorsque connecté à un autre Mac).

En plus d'utiliser l'EFI en lieu et place du Bios, Intel encourage également grandement à ce que l'EFI supporte le GPT. GPT est l'abréviation de GUID Partition Table. GUID étant également l'abréviation de Global Unique Identifier.

Apple, qui utilisait précédemment l'OpenFirmware et l'APT (son propre système de Table de Partition, Apple Partition Table) n'a eu aucun problème à adopter le GPT au lieu du MBR, vu que l'APT est plus proche, conceptuellement, du GPT que du MBR. Le choix du GPT au lieu du MBR fut donc tout à fait logique.

Mais attention. La spécification du GPT possède quelques coins d'ombre concernant sa possible implémentation. Par exemple, la GPT ne donne pas de règle stricte à propos du partitionnement d'un disque.

Apple a donc établi sa propre politique quant au partitionnement d'un disque ([Lien113](#)). Par exemple, une partition doit être alignée sur un bloc de 4K. Sur des disques compris entre 1 et 2Go (des clés usb, par exemple), une zone de 128Mo sépare les 2 partitions. Sur des disques de plus de 2Go (cad tous les disques durs vendus aujourd'hui) il y a toujours une partition réservée à l'EFI qui est créée. La taille de cette partition est fixée à 200Mo.

A noter que le firmware inclut dans les MacIntel contient également une émulation du Bios. C'est grâce à cela, que les MacIntel peuvent proposer Windows XP ou Windows Vista en Dual Boot.

### WWDC08 du 9 au 13 juin

Apple vient de publier les dates de la prochaine World Wide Developer Conference 2008, autrement dit la grande messe annuelle des développeurs Mac qui aura lieu du 9 au 13 juin 2008.

Cette édition sera classée en trois catégories principales qui sont l'iPhone, le Mac et la Technologie de l'information, et devrait annoncer la sortie finale du SDK ainsi que le lancement de l'AppStore.

Rendez-vous sur le site d'Apple Dev ([Lien114](#)) pour l'inscription (prix de l'entrée : 1295\$)

Retrouvez ces billets et de nombreux autres sur le récapitulatif des billets Mac : [Lien115](#)

# Les livres Mac

## Mac OS X Leopard Edition: The Missing Manual

### Critique par la rédaction (Vincent Brabant)

S'il n'y avait qu'un seul livre à acheter, et que vous avez les moyens, c'est celui-ci qu'il vous faut acheter.

Il est vraiment très complet, couvre tout ou presque. Est très agréable à lire.

J'ai vraiment eu l'impression d'en avoir eu pour mon argent, une fois que j'avais fermé la dernière page du livre.

Le seul reproche qu'on peut lui faire est qu'il est en Anglais. Mais cela on le sait, avant de l'acheter.

(Mais la traduction française ne devrait plus trop tarder pour ceux qui ne peuvent vraiment pas absorber près de 1000 pages.)

L'autre reproche est que certains chapitres du livre ne sont disponible qu'en ligne.

Or, si j'achète un livre en version papier, c'est justement parce que je préfère lire sur du papier que lire à l'écran. Certains trouveront peut-être bien cette idée.

Mais moi, je n'aime pas. D'où le 4.5 au lieu du 5 étoiles.

Remarque que j'ai vu récemment que la 3ième édition de ce livre était sortie, et a été mise à jour pour tenir compte des modifications apportées à Leopard 10.5.2.

Vérifiez donc bien cela lors de l'achat de ce livre en librairie.

## Mon Mac & moi : Mac OS X 10.5

Mac OS X 10.5, nom de code Leopard, intègre plus de trois cents innovations qui en font le système d'exploitation le plus abouti et le plus élégant.

Cette nouvelle version confirme la longueur d'avance prise par Mac OS X sur les systèmes d'exploitation concurrents. Sa simplicité d'utilisation, la richesse de ses fonctionnalités, sa stabilité, sa résistance aux virus et autres logiciels malveillants ainsi que son ouverture vers les technologies les plus récentes demeurent inégalées.

Vous accompagner dans l'installation, la prise en main et la personnalisation de Mac OS X 10.5, vous montrer les technologies qui vous rendront plus efficace, vous faire découvrir les applications qui vous seront les plus utiles au quotidien et vous expliquer comment réaliser un certain nombre de tâches courantes, telle est l'ambition de ce nouveau livre de la collection Mon Mac & Moi.

Mon Mac & Moi : Mieux comprendre et Mieux utiliser.

Une collection d'ouvrages ludiques, accessibles et tout en couleurs, permettant d'être rapidement efficace sur les fonctionnalités multiples et diverses de votre Mac. Tous les titres sont développés par des formateurs professionnels que vous pouvez mieux connaître en vous connectant sur le site officiel de

la collection : [www.monmacetmoi.com](http://www.monmacetmoi.com).

Vous y trouverez également les informations sur les prochaines publications ainsi que sur notre réseau de revendeurs.

### Critique par la rédaction (Vincent Brabant)

Ce livre destiné à un public de débutant est vraiment très agréable à lire.

Cela doit être du au fait qu'il est en couleur, ce qui a permis aux auteurs de ce livre d'utiliser des codes de couleurs pour la signalétique utilisé tout au long de ce livre. Ce qui facilite le repérage des astuces, remarques, dangers sur lesquels les auteurs désirent attirer notre attention.

Il y a deux, trois choses que je reproche à ce livre, qui ont quelque peu gâché mon plaisir lors de sa lecture :

- Les quelques photos (j'en ai compté que 5) d'écran faites lors de l'installation de Leopard sont bien trop foncées à mon goût que pour être correctement lisibles
- La taille de la police de caractère expliquant certaines illustrations m'a semblé quelques fois plus petite que d'habitude, au point de me gêner pour la lecture. Mais peut-être que mes yeux se font vieux.
- Pour un livre qui se veut être clair et avoir une bonne approche pédagogique, le tableau de la page 12 ne fut pas évident à comprendre. D'ailleurs je ne suis toujours pas sûr de l'avoir compris.

Mais, elles sont somme toutes mineures, comparé à la qualité d'ensemble du livre.

Quelques fois, le fait d'avoir un livre en couleur n'apporte pas toujours de plus, excepté concernant le prix de vente du livre. Ici, les auteurs ont vraiment tiré avantage du fait que le livre était imprimé en couleur. Et cela rend la lecture beaucoup plus agréable.

Le dernier chapitre est consacré à des cas pratiques comme la connexion à Internet, l'installation d'une imprimante,

...

Maintenant, il est évident qu'en 165 pages, ils ne peuvent pas tout couvrir. Mais le choix qu'ils ont fait m'a semblé très judicieux. Je dois avouer que je n'ai pas eu l'impression de rester sur ma faim une fois le livre terminé. Aussi, le fait qu'il est fortement illustré et que les illustrations soient agréablement commentées rehausse encore l'impression globale que j'ai eue de ce livre.

Pour terminer, je rappellerai qu'il s'adresse bien à un public de débutants, découvrant Mac OS X pour la première fois.

Je ne pense pas que ce livre intéressera les utilisateurs des versions précédentes de Mac OS X Leopard.

Je pense qu'ils le trouveront quelque peu ennuyant.

Retrouvez ces critiques sur la page livres Mac : [Lien116](#)

## Les derniers tutoriels et articles

### Comment éviter les duplications de code : le principe DRY (Do not Repeat Yourself)

Cet article présente le principe de programmation DRY (Do not Repeat Yourself - Ne vous Répétez pas), examine les principales causes de duplication de code ou plus généralement d'informations, et propose divers outils pour y remédier.

#### I. Introduction

Les duplications de code et plus généralement d'informations posent de sérieux problèmes dans les développements de logiciels professionnels qui doivent être maintenus pendant une longue période : ces duplications rendent les futures évolutions plus risquées et peuvent causer des bugs très difficiles à identifier.

Un développeur professionnel devrait par conséquent être familier avec le principe DRY (Do Not Repeat Yourself), lequel se traduirait en français par "Ne vous répétez pas". Ce principe a été rendu populaire par le livre *The Pragmatic Programmer* ([Lien117](#)) de Andrew Hunt et David Thomas.

D'autre part ce principe est essentiel dans le Développement Dirigé par les Tests (TDD) car il constitue le principe directeur de la phase de remaniement de code (refactoring) qui vient juste après l'écriture de code permettant d'obtenir la barre verte (voir par exemple mon précédent tutoriel ([Lien118](#))).

Enfin, il me semble que respecter ce principe est l'acte de conception le plus simple qu'un développeur débutant puisse apprendre, avant même de s'intéresser par exemple à certains patrons de conception.

Mais la principale difficulté de ce principe est qu'il paraît assez évident, si bien que l'on trouve peu d'informations complémentaires ou d'illustrations concrètes. Ce principe est souvent recommandé dans les blogs des développeurs ou des consultants, mais les commentaires restent la plupart du temps de haut niveau, comme par exemple celui de Karl Sequin ([Lien119](#)) récemment :

"Les duplications de code peuvent causer de forts maux de tête aux développeurs. Non seulement elles rendent le code plus difficile à changer (parce que vous devez trouver tous les endroits qui font la même chose), mais elles ont aussi le potentiel d'introduire de sérieux bugs et rendre la vie inutilement compliquée aux nouveaux développeurs. En suivant le principe Ne vous Répétez Pas durant toute la vie d'un système (histoires d'utilisateur, conception, codage, tests unitaires et documentation) vous arriverez à du code plus propre et plus maintenable. Gardez à l'esprit que le concept va plus loin que le copier/coller, et vise à éliminer les duplications de fonctionnalité/comportement sous toutes les formes. L'encapsulation des objets et du code très cohésif peut nous aider à réduire les duplications."

**Pour information**, le texte original : "Code duplication can cause developers major headaches. They not only make it harder to change code (because you have to find all the places that do the same thing), but also have the potential to introduce serious bugs and make it unnecessarily hard for new developers to jump onboard. By following the Don't Repeat Yourself (DRY) principal throughout the lifetime of a system (user stories, design, code,

unit tests and documentation) you'll end up with cleaner and more maintainable code. Keep in mind that the concept goes beyond copy-and-paste and aims at eliminating duplicate functionality/behavior in all forms. Object encapsulation and highly cohesive code can help us reduce duplication."

De plus il y a peu d'informations en français sur la question. Il me semble donc utile de chercher à illustrer concrètement ce principe DRY, notamment avec des exemples de code. La première section de cet article reprend et reformule la présentation originale de Hunt et Thomas. La deuxième section reprend les causes de duplications également identifiées par Hunt et Thomas. Puis des exemples concrets basés sur mon expérience personnelle sont introduits dans la troisième section. J'ai choisi d'utiliser plusieurs langages de programmation (C#, Python, Delphi) afin d'insister sur le fait que ce principe est universel.

On pourrait peut-être traduire cet acronyme **DRY** par **SEC** en français : **Surtout Evitez les Copies**

#### 2. Les bases du principe DRY

Le principe est souvent connu comme une interdiction de dupliquer du code. Mais en relisant la présentation faite dans *The Pragmatic Programmer* on se rend compte qu'il est beaucoup plus général. Les auteurs (Andrew Hunt et David Thomas) constatent qu'en tant que développeurs nous manipulons de la connaissance : nous travaillons sur sa collecte, son organisation, sa modélisation, sa maintenance, la faisons évoluer. Nous la documentons sous forme de spécifications, sous forme de commentaires dans le code, sous forme de diagrammes. Nous la rendons vivante sous forme de code exécutable, et nous l'utilisons pour les tests.

Toutefois cette connaissance que nous manipulons a une propriété remarquable : elle n'est pas stable. Elle change, souvent rapidement. Les besoins utilisateurs évoluent, des changements de réglementation doivent être pris en compte, le marché visé évolue lui aussi (par exemple à cause d'une nouvelle version produite par un concurrent, ou encore une fusion entre des concurrents qui change la donne). Les tests peuvent montrer qu'un algorithme n'est pas adapté. Notre compréhension de cette connaissance évolue au fur et à mesure de nos travaux, et nous souhaitons remanier le code, améliorer notre modèle métier.

Cette instabilité implique que nous passons une bonne partie de notre temps dans un mode de maintenance, à réorganiser cette connaissance. Il est illusoire de penser qu'une application entre en phase de maintenance après sa livraison : les développeurs sont plutôt constamment dans un mode de maintenance. Quand vous retouchez ou améliorez le code écrit hier, vous êtes déjà en train de le maintenir.

Réfléchir sur ce caractère omniprésent de la maintenance conduit

à s'apercevoir qu'une grande partie de notre activité consiste à trouver et changer la représentation d'éléments de connaissances disséminés dans un logiciel. Malheureusement il est très facile de dupliquer de la connaissance dans les spécifications, processus, programmes, et tests que nous écrivons - et la maintenance peut devenir un cauchemar bien avant que l'application ne soit livrée.

Le respect du principe DRY devrait permettre d'éviter ce problème. Il n'est pas focalisé sur les duplications de code, mais sur les duplications de connaissances :

**Principe DRY : chaque élément de connaissance doit avoir une représentation UNIQUE, NON AMBIGUË, OFFICIELLE dans un système**

Pour bien le comprendre on peut imaginer un cas où il n'est pas respecté : une chose est représentée à deux ou trois endroits différents. Si vous en modifiez un, vous devez vous rappeler de changer les autres. Pour Hunt et Thomas, la question n'est pas de savoir si vous vous rappellerez, mais de savoir quand vous oublierez.

### 3. Les causes de ces duplications

Andrew Hunt et Dave Thomas ont identifié quatre causes de duplications :

#### 3.1. Les duplications imposées

Les développeurs ont le sentiment que la duplication leur est imposée par l'environnement. Par exemple une structure de classes doit refléter un schéma de base de données. Il y a toutefois souvent des solutions pour éviter la duplication de connaissances. Dans le cas de la base de données, il est possible de générer la structure des classes depuis le schéma de la base de données. Ainsi les solutions tournent souvent autour de l'utilisation d'un générateur de code à partir d'une représentation unique de connaissances. Par contre il est vital de rendre le processus actif, dans le sens où la génération doit pouvoir être refaite à volonté ; sinon l'on retomberait dans la duplication.

Les commentaires dans le code constituent un autre cas de duplication qui paraît imposée (par exemple par les normes de codage). Mais ce problème disparaît de lui-même si les commentaires sont de bonne qualité : ils doivent éviter de paraphraser le code, et doivent comporter des explications de haut niveau qu'il n'est pas possible de déduire de la simple lecture du code. Ainsi on évitera de devoir modifier les commentaires à chaque modification du code, et on évitera que les commentaires ne se désynchronisent du code.

Devinette : qu'est-ce qui est pire que l'absence de commentaires ? C'est la présence de commentaires obsolètes. Le code lui-même ne doit pas avoir besoin de commentaires pour être compris : il doit être auto-décrit, par l'utilisation de noms de variables et de fonctions très claires, et il doit avoir une complexité réduite.

Maintenir une parfaite synchronisation entre documentation et code est souvent difficile. Là aussi il faut autant que possible générer la documentation à partir du code.

Divers outils le permettent : pour Visual Studio, voir par exemple GhostDoc ([Lien120](#)) qui permet de générer automatiquement les entêtes de commentaires XML. Parfois il est possible de générer directement certains tests à partir d'un document de spécification. Voir notamment FitNess ([Lien121](#)) qui permet de rédiger des tests sous forme de tables dans les pages html d'un wiki, puis de les exécuter et de présenter les résultats dans le wiki. L'apparition

récente d'un outil commercial comme GreenPepper ([Lien122](#)) (permettant de rédiger des "spécifications exécutable") est sans doute une indication de la pertinence des principes proposés initialement par FitNess.

Enfin certains types de duplications imposées peuvent provenir directement du langage de programmation, qui parfois oblige à dupliquer les signatures de fonctions entre une partie interface et une partie implémentation (C++, Delphi). Il n'y a alors pas grand chose à faire, mais ici l'inconvénient est réduit du fait que le compilateur indiquera les éventuelles incohérences.

#### 3.2. Les duplications par inadvertance

Les développeurs ne réalisent pas qu'ils sont en train de dupliquer de l'information. Il s'agit ici essentiellement d'erreurs dans la conception, quand deux classes contiennent le même élément d'information. D'autre part, une classe définit parfois des champs mutuellement dépendants ; dans ce dernier cas il est sans doute préférable de remplacer l'un des champs par une méthode réalisant le calcul à chaque accès.

#### 3.3. Les duplications par impatience

Les développeurs dupliquent par paresse ou par facilité ou encore sous la pression de dates de livraison à respecter. Très souvent, nous préférons dupliquer une méthode, une classe, un fichier plutôt que de prendre le temps (et peut-être le courage) de factoriser proprement les éléments correspondants. Cette tendance naturelle est de plus encouragée par les contraintes de temps, et surtout par l'absence de tests automatisés. En effet, dupliquer permet de modifier une seule zone de code, alors que factoriser implique nécessairement de modifier au moins deux zones ; en l'absence de tests automatisés, le risque apparaît alors trop grand. Malheureusement le principal remède ici est l'autodiscipline et la volonté de passer plus de temps maintenant pour en économiser plus tard.

Un lecteur ([olsimare Lien123](#)) me signale très justement que ce type de duplication arrive dans le domaine des bases de données, et **au niveau des données elles-mêmes**. Par impatience, ou paresse, ou peur des risques, on duplique parfois des données au lieu de faire le travail plus difficile de restructuration qui aurait été nécessaire. Au fil du temps et des éventuels problèmes, la cohérence entre les données dupliquées devient de plus en plus difficile à maintenir.

#### 3.4. Les duplications inter-développeurs

Cela arrive quand des développeurs codent indépendamment la même fonctionnalité. Normalement une architecture de logiciel bien définie et claire devrait réduire ce risque, mais il y a toujours des fonctionnalités communes (comme des fonctions utilitaires) qui n'appartiennent pas clairement à tel ou tel élément de l'architecture. Il y a alors de grandes chances que ces utilitaires soient développés plusieurs fois. Le remède ici est de centraliser les utilitaires, de faciliter la communication entre développeurs (forums, wiki, ...), et de se forcer à lire le code de ses camarades. Le travail en duo (pair-programming) recommandé par XP apporte également des pistes de solutions.

## 4. Exemples

### 4.1. Synchronisation de code et de document

Quand on cherche à maintenir l'**unicité** des éléments de connaissances alors que l'on a besoin de cette connaissance sous différents formats, il faut utiliser un processus de génération automatique pour obtenir ces formats à partir de la représentation

unique.

Les extraits de code que l'on inclut dans un livre ou un article sont un cas assez courant où un tel processus est très utile. Sans processus automatique, à chaque modification du code exemple, il faut penser à modifier le livre ou l'article, rechercher l'endroit où propager la modification, et éventuellement adapter la mise en page de ce que l'on vient de dupliquer. Tout cela est évidemment source d'erreurs, de travail supplémentaire, et de fatigue due au caractère inintéressant de ces opérations.

Récemment certains auteurs vont encore plus loin : le livre xUnit Test Patterns ([Lien124](#)) de Gerard Meszaros a d'abord existé sous la forme d'un site Web, et l'auteur a développé toute une série de scripts Ruby pour pouvoir générer son livre à partir du contenu du site Web. Par la suite, seul le livre a évolué, ce qui rend le site Web de plus en plus obsolète - c'est donc une violation du principe DRY, mais qui a été faite volontairement afin de favoriser la vente du livre !

Ici j'illustre ce même principe (à une échelle beaucoup plus réduite), en montrant comment je vais inclure des exemples de code dans cet article à l'aide d'un tel processus automatisé. Heureusement les articles publiés sur ce site sont des fichiers xml, ce qui facilite grandement leur manipulation par un programme séparé.

La première étape de mon processus est d'inclure une balise code dans mon fichier xml :

```
<code langage="python"
autoinsert="F:\\DEV\\python\\remplace-code-dans-
xml.py::signet-dvp-1">
    --- le code sera inséré ici ---
</code>
```

La deuxième étape consiste à exécuter le script python sur mon document xml. Le script recherche simplement les noeuds "code", extrait les lignes de code situées entre les signets dans les fichiers indiqués par l'attribut "autoinsert", modifie les noeuds "code", et régénère le fichier xml (en ayant pris soin d'en faire une copie). Voici le script en question :

```
# Pour fonctionner sur des fichiers contenant des
caractères accentués,
# ce script nécessite la présence des lignes
# import sys
# sys.setdefaultencoding('latin-1')
# dans le fichier Python/Lib/site-
packages/sitecustomize.py
# reference :
http://personalpages.tds.net/~kent37/blog/stories/14.ht
ml

from __future__ import with_statement
import os
import sys
import shutil
import xml.etree.ElementTree
from xml.etree.ElementTree import ElementTree

def traitementArticle(fichier):
    shutil.copyfile(fichier, nomFichierUnique(fichier))
    tree = ElementTree(None, fichier)
    map(insertionCodeDansBalisesXML, tree.getiterator('c
ode'))
    tree.write(fichier, 'iso-8859-1')
    remplacementChr160DansFichier(fichier)

def insertionCodeDansBalisesXML(element):
```

```
try:
    (fichier, signet)=element.attrib['autoinsert'].s
plit('::')
    element.text =
extractionLignesEntreSignets(fichier, signet)
except:
    print "erreur en traitant: ", element.attrib
    print sys.exc_info()

def extractionLignesEntreSignets(fichier, signet):
    resultat=""
    with open(fichier) as f:
        for ligne in f:
            if ligne.find(signet)>=0:
                if resultat != "":
                    break
                else:
                    resultat = "\n"
                    continue
            if resultat != "":
                resultat += ligne
    if resultat == "":
        resultat = "signet " + signet + "non trouvé"
    return resultat

def fabriqueNomFichier(nom, index):
    return nom + "." + str(index)

def nomFichierUnique(fichier):
    index = 1
    while
(os.path.exists(fabriqueNomFichier(fichier, index))):
        index += 1
    return fabriqueNomFichier(fichier, index)
```

```
# J'ai constaté que la représentation "&#160;" du
caractère "espace insécable"
# de l'éditeur XML était remplacée par le caractère de
code 160 durant la
# transformation. Pour l'instant je n'ai pas trouvé
d'autre solution que de réouvrir
# le fichier pour remplacer le caractère 160 par la
bonne représentation.
def remplacementChr160DansFichier(fichier):
    with open(fichier) as f:
        s = f.read().replace(chr(160), "&#160;")
    with open(fichier, 'w') as f:
        f.write(s)

if __name__ == "__main__":
    traitementArticle('C:\\Article_Dvp\\documents\\prin
cipe_dry\\principe_dry.xml')
```

Comme l'attribut "autoinsert" n'est pas modifié dans ce processus, il est possible de recommencer autant de fois que nécessaire - ce qui est une propriété absolument indispensable pour éviter de retomber dans de futures duplications. Ma première idée était plutôt du type :

```
<code langage="python">
autoinsert="F:\\DEV\\python\\remplace-code-dans-
xml.py::signet-dvp-1"
</code>
```

mais l'information sur "autoinsert" était perdue lors du remplacement par le code, et donc la propriété désirée n'était pas respectée.

Le script ci-dessus tente d'éviter des duplications : en particulier les noms de fonctions très explicites, les fonctions très courtes et

de complexité réduite évitent de commenter le code. Ainsi les commentaires sont limités aux informations qu'il est impossible de déduire du code lui-même, et se focalisent sur le **Pourquoi** au lieu de dupliquer le **Comment**.

Toutefois la fonction `extractionLignesEntreSignets` n'est peut-être pas suffisamment facile à comprendre. Un commentaire pour expliquer ce qu'elle fait pourrait alors sembler utile : ignorer les lignes avant et après le signet, concaténer les lignes entre les deux occurrences du signet... Mais dans une optique DRY il est plutôt souhaitable de la réécrire afin de faciliter sa lecture, et sa future maintenance. Je laisse la réécriture en exercice pour les lecteurs !

Il se trouve que ce petit script Python illustre une des autres recommandations de *The Pragmatic Programmer* ([Lien117](#)) : apprenez un langage de manipulation de texte. Hunt et Thomas appellent ainsi les langages tels que Python, Perl, Ruby, qui permettent à un développeur d'automatiser diverses tâches répétitives et ainsi de pallier les limitations de son environnement de développement. J'adhère sans réserve à cette recommandation : si vous ne pratiquez pas de ces langages, apprenez-en un ! Vous pourriez être surpris de découvrir que cela vous rend service tous les jours. Et apprendre Python ou Ruby est l'affaire de quelques heures. Dans un contexte Windows/.NET, PowerShell ([Lien125](#)) est également une option.

Pour moi le bilan de ce petit script est très positif : n'ayant encore jamais manipulé de XML avec Python, 1 heure de recherche et d'expérimentation m'a été nécessaire pour déterminer le cœur du script, à savoir l'obtention d'un itérateur sur toutes les balises "code". Cette heure est un bon investissement car manipuler des fichiers XML est un besoin fréquent pour un programmeur, et cela me rassurera sans aucun doute. J'ai ensuite perdu beaucoup plus de temps à résoudre les questions d'encodages de caractères mentionnées dans les commentaires, mais le confort que m'apporte maintenant ce script vaut largement le temps passé. Mon seul regret est de ne pas avoir fait plus tôt ce petit effort, cela m'aurait évité du travail de synchronisation bien désagréable sur de précédents articles où mon seul outil était Copier/Coller.

#### 4.2. Exemple de duplication par inadvertance

Voici maintenant un exemple tiré d'un cas réel, où le non-respect du principe DRY a entraîné un défaut sérieux. A l'origine, un développeur A (moi-même en l'occurrence, il y a bientôt dix ans) doit afficher un signal, et doit le normaliser avant de l'afficher, c'est-à-dire ramener toutes les valeurs dans l'intervalle [0;100%]. Le développeur A programme donc le code suivant :

```
public class Afficheur
{
    private Donnees donnees;
    private Graphique graphique = new Graphique();

    public Afficheur(Donnees donnees)
    {
        this.donnees = donnees;
    }

    public void Affiche()
    {
        double pasX = 0 ;
        foreach (double donnee in donnees)
        {
            double valeur_normalisee = 100.0 * (donnee
- donnees.Minimum) / (donnees.Maximum -
donnees.Minimum);
            pasX += 0.1;
            graphique.AddXY(pasX, valeur_normalisee);
        }
    }
}
```

```
}
}
```

Pour faire au plus simple et au plus rapide, le développeur A a fait le calcul de normalisation "au vol" et au dernier moment. Ce choix qui paraissait judicieux à l'époque du développement initial s'est révélé lourd de conséquences durant l'ajout de fonctionnalités supplémentaires (comme l'impression du graphique).

Pour information, ce code s'appuie sur le squelette de classes suivant (j'ai écrit le strict minimum pour parvenir à compiler) :

```
public class Graphique
{
    public void AddXY(double x, double y)
    {
        // ...
    }
}

public class Donnees : IEnumerable
{
    double minimum ;
    double maximum ;
    private ArrayList donneesInternes = new
ArrayList();

    public Donnees (ArrayList donnees)
    {
        minimum = Double.MaxValue;
        maximum = - Double.MaxValue;
        foreach (double d in donnees)
        {
            minimum = Math.Min(minimum, d);
            maximum = Math.Max(maximum, d);
            donneesInternes.Add(d);
        }
    }

    public double this[int pos]
    {
        get { return (double)donneesInternes[pos]; }
    }

    public IEnumerator GetEnumerator()
    {
        return donneesInternes.GetEnumerator();
    }

    public double Maximum
    {
        get { return maximum;}
    }

    public double Minimum
    {
        get { return minimum;}
    }
}
```

Un an plus tard, un développeur B doit programmer un algorithme de calcul qui a besoin de plusieurs méthodes de normalisation. Malheureusement il ne touche pas au code de normalisation qui existe déjà dans la classe `Afficheur`. En effet,

- soit il ne connaît pas la classe `Afficheur`, et ne sait donc pas que cette méthode existe,
- soit il la connaît, mais il n'ose pas y toucher. Peut-être parce qu'il n'a pas le temps, ou encore parce que il n'ose pas toucher à du code existant (il n'a pas de tests automatisés qui lui permettraient de remanier le code en sécurité)

- soit il estime qu'il n'a pas à y toucher, car aucune des méthodes de normalisation qui l'intéressent n'est exactement la même que celle de la classe Afficheur

Quelle que soit la raison, le développeur B développe sa propre bibliothèque de plusieurs méthodes de normalisation et ne touche pas au code existant.

Deux ans plus tard, un développeur C doit réaliser l'impression de l'écran affiché initialement par le développeur A. Le moteur d'impression est totalement indépendant de la classe Afficheur, il n'est pas possible d'utiliser le même composant Graphique, et il faut en gros afficher les mêmes données sur un autre support (ce qui en soit est aussi une forme de duplication). Développeur C n'a donc pas accès au calcul de normalisation noyé dans la classe Afficheur, et s'oriente assez logiquement vers la bibliothèque de développeur B. Il y choisit une des méthodes fournies et l'utilise pour réaliser l'impression.

Hélas l'équivalent de la méthode contenue dans Afficheur.Affiche n'était pas dans la bibliothèque, et celle choisie donne à peu près les mêmes résultats sur certains jeux de données, en particulier ceux utilisés dans les tests manuels. Et ensuite pendant plusieurs années les résultats imprimés sont différents des résultats affichés. Mais cela passe inaperçu jusqu'à ce qu'un nouveau type de données chez un client rende la différence apparente.

Développeur A écrit code de normalisation directement dans GUI

```
public void Affiche()
{
    double pasX = 0;
    foreach (double donnee in donnees)
    {
        double valeur_normalisee = 100.0 * (donnee - donnees.Minimum) / (
            donnees.Maximum - donnees.Minimum);
        pasX += 0.1;
        graphique.AddXY(pasX, valeur_normalisee);
    }
}
```

Développeur C réalise l'impression et prend une des fonctions de la bibliothèque – pas la bonne car elle n'est pas dans la bibliothèque

Bug critique non détecté pendant plusieurs années car les résultats sont souvent proches

Développeur B écrit bibliothèque d'outils de normalisation – mais ne touche pas au code existant → duplication

de point de départ :

```
[TestFixture]
public class TestNormalisation
{
    [Test]
    public void VerificationValeursAttendues()
    {
        ArrayList donnees_test = new ArrayList();
        donnees_test.Add(1.0);
        donnees_test.Add(2.0);

        Donnees donnees = new Donnees(donnees_test);

        Normalisateur norm = new
Normalisateur(donnees);

        Assert.AreEqual(0.0, norm[0], 0.0001);
        Assert.AreEqual(1.0, norm[1], 0.0001);
    }
}
```

On voit que ce test fait appel à une nouvelle classe Normalisateur, qui traduit notre souci d'extraire la méthode de calcul afin de la rendre testable. Une autre option aurait été d'ajouter cette méthode de calcul directement à la classe Donnees, mais nous faisons volontairement ce choix de conception afin de limiter les responsabilités de la classe Donnees.

```
public class Normalisateur : IEnumerable
{
    private Donnees donnees;

    public Normalisateur(Donnees donnees)
    {
        this.donnees = donnees;
    }

    public double this[int pos]
    {
        get { return 100.0 * (donnees[pos]-
donnees.Minimum)/(donnees.Maximum - donnees.Minimum); }
    }

    public IEnumerator GetEnumerator()
    {
        return donnees.GetEnumerator();
    }
}
```

Bien sûr l'introduction de la classe Normalisateur a des conséquences sur notre Afficheur, mais ces conséquences sont positives car Afficheur dépend maintenant uniquement d'une interface IEnumerable. Nous avons donc réduit le couplage de nos classes.

```
public class AfficheurTDD
{
    private IEnumerable donnees;
    private Graphique graphique = new Graphique();

    public AfficheurTDD(IEnumerable donnees)
    {
        this.donnees = donnees;
    }

    public void Affiche()
    {
        double pasX = 0;
        foreach (double donnee in donnees)
```

La morale de cet exemple, simplifié mais réel, est qu'à force de ne pas traiter les duplications au fur et à mesure que l'on ajoute du code, on introduit de la confusion et des problèmes très difficiles à repérer. Cet exemple montre aussi pourquoi ces duplications ne sont pas traitées. D'une part, en l'absence de tests automatisés, un développeur préfère ne pas prendre de risque et préfère dupliquer plutôt que modifier du code existant. D'autre part, traiter les duplications peut exiger des remaniements qui apparaissent trop importants, et trop coûteux, au vu de l'état actuel de l'architecture du logiciel : c'est le cas si l'on avait voulu s'affranchir d'un support particulier pour pouvoir utiliser la même méthode d'affichage à l'écran que pour l'impression - cela remettait en cause l'architecture suivie jusque-là.

Ici on peut noter que si développeur A avait travaillé en TDD, le problème aurait pu être éliminé à la source. En effet travailler en TDD aurait obligé à extraire la méthode de normalisation de la classe Afficheur (afin de la rendre testable) et aurait donc eu pour conséquence bénéfique de la rendre disponible aux développeurs suivants. Développeur B aurait donc été incité à l'inclure dans sa bibliothèque, et développeur C aurait eu le choix de la bonne méthode.

Afin de terminer notre exemple, voici comment le problème aurait pu être traité en TDD. Tout d'abord voici un des tests qui servirait

```
{  
    pasX += 0.1;  
    graphique.AddXY(pasX, donnee);  
}  
}
```

## 5. Conclusion

Avec mon expérience de développement et de maintenance d'un logiciel pendant près de 10 ans, je constate assez fréquemment les problèmes difficiles posés par la duplication de code ou de connaissances. En réalité, plus que la duplication elle-même, c'est plutôt la duplication erronée ou inconsistante qui pose problème en pratique. Soit la duplication était erronée dès le départ, soit le code dupliqué évolue de façon inconsistante à plusieurs endroits. Cela rend d'autant plus difficile la recherche automatisée de

duplications. Les rares outils de recherche de duplication comme Simian - Similarity Analyser (commercial) ([Lien126](#)) ou CPD ([Lien127](#)) (gratuit) semblent surtout efficaces pour identifier le code dupliqué à l'identique, et butent sur les duplications inexactes.

Hélas il n'y a pas de solution bien simple, et la première action possible est probablement la sensibilisation des développeurs aux risques cachés derrière la duplication, afin qu'ils prennent le temps de réfléchir au moment où ils font Copier/Coller. Avec cet article j'espère avoir contribué à cette sensibilisation. J'ai également indiqué divers outils qui peuvent faciliter la vie des développeurs soucieux de respecter le principe DRY.

Retrouvez l'article de Bruno Orsier en ligne : [Lien128](#)

# Liens

- Lien1 : <http://dico.developpez.com/html/3107-Langages-JSF-Java-Server-Faces.php>  
Lien2 : <http://dico.developpez.com/html/2990-Langages-EDI-Environnement-de-Developpement-Integre.php>  
Lien3 : <http://dico.developpez.com/html/3038-Langages-Eclipse.php>  
Lien4 : <http://djo-mos.developpez.com/tutoriels/java/jsf/Eclipse-wtp2-config/>  
Lien5 : <http://dico.developpez.com/html/865-Langages-Classpath.php>  
Lien6 : <http://java.sun.com/javaee/javaserverfaces/download.html>  
Lien7 : <http://dico.developpez.com/html/903-Langages-servlet.php>  
Lien8 : <http://dico.developpez.com/html/3020-Conception-MVC-Model-View-Controller.php>  
Lien9 : <http://download.java.net/javaee5/promoted/shared/glassfish-persistence/glassfish-persistence-installer-v2-b41.jar>  
Lien10 : <http://dico.developpez.com/html/156-Langages-JDBC-Java-DataBase-Connectivity.php>  
Lien11 : [http://djo-mos.developpez.com/tutoriels/java/crud-jsf-jpa/#IMPORTANT\\_WARNING](http://djo-mos.developpez.com/tutoriels/java/crud-jsf-jpa/#IMPORTANT_WARNING)  
Lien12 : [http://djo-mos.developpez.com/tutoriels/java/crud-jsf-jpa/#IMPORTANT\\_WARNING2](http://djo-mos.developpez.com/tutoriels/java/crud-jsf-jpa/#IMPORTANT_WARNING2)  
Lien13 : <http://djo-mos.developpez.com/tutoriels/java/crud-jsf-jpa/>  
Lien14 : <http://php.developpez.com/cours/?page=bibliotheques#com>  
Lien15 : <http://search.cpan.org/~jmcnamara/Spreadsheet-WriteExcel-2.20/lib/Spreadsheet/WriteExcel.pm>  
Lien16 : <http://search.cpan.org/~jmcnamara/>  
Lien17 : <ftp://ftp-developpez.com/g-ernaelsten/sources.zip>  
Lien18 : [http://www.bettina-attack.de/jonny/view.php/projects/php\\_writeexcel/](http://www.bettina-attack.de/jonny/view.php/projects/php_writeexcel/)  
Lien19 : [http://pear.php.net/package/Spreadsheet\\_Excel\\_Writer](http://pear.php.net/package/Spreadsheet_Excel_Writer)  
Lien20 : <http://pear.php.net/manual/fr/package.fileformats.spreadsheet-excel-writer.php>  
Lien21 : <http://pear.php.net/>  
Lien22 : <http://php.developpez.com/exemples/excel/fichierBase.php>  
Lien23 : <http://php.developpez.com/exemples/excel/example-bigfile.php>  
Lien24 : <http://php.developpez.com/exemples/excel/woorksheet.php>  
Lien25 : <http://php.developpez.com/exemples/excel/example-formatMerge.php>  
Lien26 : <http://php.developpez.com/exemples/excel/example-formatRotation.php>  
Lien27 : <http://g-ernaelsten.developpez.com/tutoriels/excelphp/>  
Lien28 : <http://a-pellegrini.developpez.com/tutoriels/xhtml-css/tableaux/fichiers/page1-td-tr.html>  
Lien29 : <http://a-pellegrini.developpez.com/tutoriels/xhtml-css/tableaux/fichiers/page2-th.html>  
Lien30 : <http://a-pellegrini.developpez.com/tutoriels/xhtml-css/tableaux/fichiers/page3-caption.html>  
Lien31 : <http://a-pellegrini.developpez.com/tutoriels/xhtml-css/tableaux/fichiers/page4-thead.html>  
Lien32 : <http://a-pellegrini.developpez.com/tutoriels/xhtml-css/tableaux/fichiers/page5-col.html>  
Lien33 : <http://a-pellegrini.developpez.com/tutoriels/xhtml-css/tableaux/fichiers/page6-colspan.html>  
Lien34 : <http://a-pellegrini.developpez.com/tutoriels/xhtml-css/tableaux/fichiers/page6-rowspan.html>  
Lien35 : <http://a-pellegrini.developpez.com/tutoriels/xhtml-css/tableaux/fichiers/page7-summary.html>  
Lien36 : <http://a-pellegrini.developpez.com/tutoriels/xhtml-css/tableaux/fichiers/page8-abbr-scope-header.html>  
Lien37 : <http://a-pellegrini.developpez.com/tutoriels/xhtml-css/tableaux/fichiers/page9-centrage.html>  
Lien38 : <http://a-pellegrini.developpez.com/tutoriels/xhtml-css/tableaux/fichiers/page10-bordure-externe.html>  
Lien39 : <http://a-pellegrini.developpez.com/tutoriels/xhtml-css/tableaux/fichiers/page11-bordure-separee.html>  
Lien40 : <http://a-pellegrini.developpez.com/tutoriels/xhtml-css/tableaux/fichiers/page12-bordure-fusionnee.html>  
Lien41 : <http://a-pellegrini.developpez.com/tutoriels/xhtml-css/tableaux/fichiers/page13-empty.html>  
Lien42 : <http://a-pellegrini.developpez.com/tutoriels/xhtml-css/tableaux/fichiers/page14-caption.html>  
Lien43 : <http://a-pellegrini.developpez.com/tutoriels/xhtml-css/tableaux/fichiers/page15-css.html>  
Lien44 : <http://a-pellegrini.developpez.com/tutoriels/xhtml-css/tableaux/>  
Lien45 : <http://cssglobe.developpez.com/tutoriels/css/bouton-redimensionnable/fichiers/buttons.htm>  
Lien46 : <http://cssglobe.developpez.com/tutoriels/css/bouton-redimensionnable/>  
Lien47 : <http://iteratif.developpez.com/articles/flex/applications-localisees/>  
Lien48 : <ftp://ftp-developpez.com/lunar/tutoriels/flex/compilation/fichiers/files.zip>  
Lien49 : <http://lunar.developpez.com/tutoriels/flex/compilation/>  
Lien100 : <http://msdn2.microsoft.com/fr-fr/library/system.web.ui.webcontrols.repeater.aspx>  
Lien101 : [http://nico-pyright.developpez.com/tutoriel/asp\\_net/csharp/templatecontrol/](http://nico-pyright.developpez.com/tutoriel/asp_net/csharp/templatecontrol/)  
Lien102 : <http://dotnet.developpez.com/livres/>  
Lien103 : <http://melem.developpez.com/langagec/objc/>  
Lien104 : <http://www.codeblocks.org/>  
Lien105 : <http://prdownloads.sf.net/mingw/MinGW-5.1.3.exe>  
Lien106 : <http://prdownloads.sf.net/mingw/gdb-6.6.tar.bz2>  
Lien107 : <http://download.berlios.de/codeblocks/codeblocks-8.02mingw-setup.exe>  
Lien108 : <http://nicolasj.developpez.com/gtk/windows/>  
Lien109 : <http://www.fs-security.com/download.php>  
Lien110 : <http://gorgonite.developpez.com/tutoriels/linux/firestarter/>  
Lien111 : <http://linux.developpez.com/livres/>  
Lien112 : <http://baptiste-wicht.developpez.com/tutoriel/windows/demarrage/>  
Lien113 : <http://developer.apple.com/technotes/tn2006/tn2166.html#SECPARTITIONINGPOLICY>  
Lien114 : <http://developer.apple.com/wwdc/>  
Lien115 : <http://blog.developpez.com/recap/mac>  
Lien116 : <http://mac.developpez.com/livres/>

Lien117 : <http://www.pragprog.com/the-pragmatic-programmer>  
Lien118 : <http://bruno-orsier.developpez.com/tutoriels/TDD/pentaminos/>  
Lien119 : <http://codebetter.com/blogs/karlseguin/archive/2007/11/26/foundations-of-programming-part-1-introduction.aspx>  
Lien120 : <http://www.roland-weigelt.de/ghostdoc/>  
Lien121 : <http://fitnesse.org/>  
Lien122 : <http://www.greenpeppersoftware.com/fr/>  
Lien123 : <http://www.developpez.net/forums/member.php?u=131013>  
Lien124 : <http://xunitpatterns.com/index.html>  
Lien125 : <http://www.microsoft.com/windowsserver2003/technologies/management/powershell/default.aspx>  
Lien126 : <http://www.redhillconsulting.com.au/products/simian/overview.html>  
Lien127 : [http://www.onjava.com/pub/a/onjava/2003/03/12/pmd\\_cpd.html](http://www.onjava.com/pub/a/onjava/2003/03/12/pmd_cpd.html)  
Lien128 : <http://bruno-orsier.developpez.com/principes/dry/>