# Developpez

## Le Mag

Edition de Juin – Juillet 2010.
Numéro 28.
Magazine en ligne gratuit.
Diffusion de copies conformes à l'original autorisée.
Réalisation : Alexandre Pottiez
Rédaction : la rédaction de Developpez
Contact : magazine@redaction-developpez.com

## Sommaire



## Editorial

Ce sont les grandes vacances et tout le monde prend du repos bien mérité mais attention à ne pas perdre les bonnes habitudes !
Ne vous inquiétez pas, Developpez.com ne les perd pas et vous propose un nouveau numéro de son magazine pour que vous ne perdiez pas la main pendant vos vacances.

La rédaction

### Article Développement Web


## Comment créer facilement un framework JavaScript – Partie 3

Apprenez pas à pas à créer un framework JavaScript.

par **Teylor Feliz**
Page 27


### Article C/C++/GTK

## Interview de James Reinders

*Dans cette interview, James Reinders, le gourou d'Intel sur le développement orienté développement parallèle et auteur d'un livre sur les Thread Building Blocks, fait le point sur les outils prévus en 2010.*

par **Loïc Joly**
Page 30

## Gradle, le nouvel outil de build incontournable

Le système de build propose une approche flexible pour la construction de projets Java, Groovy et Scala, et Java EE. Véritable alternative à Ant et Maven, il est aussi capable de s'intégrer à ses deux concurrents.

### 1. Introduction

La réalisation d'une application est constituée de nombreuses étapes. Parmi celles-ci, l'étape de construction, aussi appelée étape de build, est la plus importante. Elle permet de transformer un code source, issu d'une représentation humaine, dans un ensemble de codes exécutables constituant une représentation machine. Il n'est pas rare que cette étape soit complexe, tant par sa mise en œuvre, que son adéquation aux besoins changeants du projet. Il est alors nécessaire d'outiller cette construction afin de répondre au plus près aux besoins du projet.

De nombreux outils de build existent pour répondre à cette problématique. Certains de ces outils, implantés depuis des années, dominent le marché comme Ant et Maven. Néanmoins, malgré leur omniprésence, nombreux sont les développeurs qui restent bloqués et souffrent d'un manque de fonctionnalités. Gradle, né de l'expérience des utilisateurs acquise au cours ces différentes années, tente de répondre aux besoins particuliers de build des applications d'une entreprise.

### 2. Un système de build de troisième génération

Gradle est un système de build de troisième génération proposant une approche flexible pour la construction de projets Java, Groovy et Scala et Java EE. Il utilise les meilleures fonctionnalités de chacun des outils existants comme une structure de développement cadrée, une ossature pour la construction de son projet, des conventions de noms, tout en proposant en permanence une personnalisation selon ses besoins. En termes d'architecture technique, Gradle est un toolkit fondé sur les meilleures librairies du marché de l'intégration continue, comme le gestionnaire de dépendances Ivy, l'utilisation native de taches Ant et le langage Groovy pour décrire le build d'un projet.
A l'image de ses concurrents, Gradle apporte une approche déclarative dans la définition de la chaîne de build. Mais Gradle fournit un langage de spécification du domaine (DSL) en langage Groovy pour écrire un script Gradle. Cette DSL orchestre une API Gradle et permet de fournir à l'écriture du build une grande richesse des éléments. Chaque script Gradle configure de manière sous-jacente un objet Gradle Projet sûr lequel le script utilise un ensemble de propriétés et de méthodes exposées par l'API.

Ce véritable langage de build vous offre une flexibilité d'expression répondant aux besoins de votre entreprise, comme la possibilité de pouvoir déclarer plusieurs répertoires de sources par projet, de configurer les dépendances très finement, de mettre en place plusieurs projets par répertoires de sources ou encore de générer autant d'artefacts que nécessaire pour votre projet.

### 3. Gradle permet de résoudre les problèmes rencontrés avec Ant ou Maven

Au-delà d'être un compétiteur face aux outils de build existants, Gradle fait office de fédérateur. En effet, l'outil peut s'insérer aisément dans des infrastructures utilisant Ant ou Maven, ceci afin d'enrichir les fonctionnalités existantes, et offre une panoplie de stratégies pour résoudre les problèmes rencontrés avec Ant ou Maven.
Dans le cadre d'Ant, Gradle possède un module d'import permettant de convertir toutes les cibles d'un script Ant en tâches Gradle. Celui-ci peut ensuite rajouter du comportement avant ou après l'exécution de la cible Ant. Cette fusion native entre Ant et Gradle permet également de faire référence à une tâche Gradle depuis un script Ant importé. Cette mutualisation donne la possibilité ainsi d'utiliser Gradle comme outil complémentaire à votre infrastructure Ant sans un besoin de migration.

Dans le cadre de Maven, Gradle est capable de collecter des artefacts dans des dépôts Maven à travers la librairie Ivy et d'écrire dans des dépôts Maven via l'utilisation native du module Maven inclut dans la distribution Gradle. Cette fonctionnalité permet d'utiliser Gradle pour ses chaînes de build et d'utiliser des dépôts d'entreprise de type Maven en standard dans la majorité des organisations. Ces dépôts, gérés par des outils gestionnaires comme JFrog Artifactory, rendent possible une centralisation du stockage des librairies utilitaires publiques issues de dépôts Internet avec les artefacts issus des projets, le tout avec des métadonnées de type uniforme.
Gradle augmente également les aspects de fiabilité et de performance d'une chaîne de compilation en fournissant un support extrêmement puissant, pour un projet multi module, à travers des fonctionnalités de build incrémentales que ses compétiteurs ne possèdent pas.

Après deux ans d'existence, Gradle atteint un niveau de maturité lui permettant d'être adopté en masse et sans risques par les grands comptes. A ce jour, les entreprises ayant utilisé Gradle ont constaté une plus value immédiate et l'ont implanté comme leur principal outil de build.

### 4. Liens

- Comparatif des outils de build pour Java (Lien2).
- Le forum d'entraide sur Gradle (Lien4).
- Le forum d'entraide sur les outils d'intégration continue (Lien5).

*Retrouvez l'article de Grégory Boissinot en ligne : Lien6*

## Interview de David Booth et Jevgeni Kabanov (ZeroTurnaround)

David et Jevgueni, de la société ZeroTurnaround, éditeur de l'outil de productivité JRebel pour le développement Java, ont accepté de répondre à nos questions sur leur société et leur produit phare.

### 1. Au sujet de ZeroTurnaround

**ZeroTurnaround**

ZeroTurnaround est un petit éditeur logiciel estonien (le pays à l'origine de Skype, Kazaa, etc). Nous construisons de petits outils qui apportent une grande valeur ajoutée aux équipes de développement (ainsi qu'à leur budget). Par exemple, JRebel est un fichier JAR qui fait moins d'un MB, facile d'installation et d'utilisation, qui économise entre trois et sept semaines de temps de développement perdus (sur la base de semaines de 40 heures) et vous coûte moins de 150 $ par an. C'est un assez gros challenge. A mesure que nous grandissons, notre vision de l'amélioration de la qualité, de la cohérence, et de la productivité dans l'industrie du logiciel grandit.

**Jevgeni Kabanov - Founder, Chief Technical Officer**

Auparavant, Jevgeni travaillait en tant que directeur R&D de Webmedia, Ltd, la plus grande société balte de développement de logiciels custom. Dans le cadre des efforts de réduction du temps de développement, il écrivit un prototype de JavaRebel et en dirige, depuis la création de ZeroTurnaround en 2007, le développement. Il est speaker depuis quelques années lors de conférences internationales, parmi lesquelles JavaPolis/Devoxx, JavaZone, et JAOO sur des sujets comme : "Do you really get Classloaders" à "Zero Turnaround in Java Development". Il a un intérêt certain pour la recherche, avec de nombreuses publications sur des sujets allant de notions théoriques jusqu'aux typesafe DSLs Java. A côté des produits commerciaux réalisés dans le cadre de ZeroTurnaround, Jevgeni est le co-fondateur de deux projets open source - Aranea et Squill. Aranea (Lien7) est une plateforme de développement Web et d'intégration basée sur de solides principes orientés objet. Squill (Lien8) est un DSL interne type-safe pour la construction et l'exécution de requêtes SQL. Le blog personnel de Jevgeni peut être trouvé sur dow.ngra.de (Lien9).

**David Booth - Chief Executive Officer**

David a été en charge du marketing d'outils de productivité dans l'industrie Java depuis 2001, en commençant par JProbe, puis pour JetBrains (IntelliJ IDEA), prêchant pour la qualité dans le logiciel et l'amélioration de la productivité des développements. Il est convaincu de la nécessité de soutenir les communautés d'utilisateurs, et a fondé sur son temps libre le Young Professionals Forum (Lien10) à Prague où il réside.

### 2. ZeroTurnaround

**Combien de salariés sont employés par ZeroTurnaround ?**

***Jevgueni*** : Un peu moins de 10 en ce moment, sachant que cela évolue au rythme du recrutement de nouvelles recrues ... d'ailleurs connaissez-vous des ingénieurs intéressés pour travailler en Estonie, ou des commerciaux et marketeux intéressés par Prague ?

**Quel EDI est principalement utilisé par les équipes de ZeroTurnaround ? Pourquoi ?**

***Jevgueni*** : Tout le monde utilise Eclipse. C'est probablement en partie de ma faute, étant donné que j'étais un fervent partisan d'Eclipse il y a quelques années. Maintenant ça m'est indifférent, mais je pense qu'Eclipse possède de meilleures fonctionnalités Java de base et nous ne faisons pas vraiment de développement Web. Nous achèterions probablement des licences IDEA si nous avions à faire du Web, mais je continue à être mécontent de leur support du Compile-On-Save et je pense qu'Eclipse a de l'avance sur cet aspect.

**Certaines sociétés refusent de payer sur la base d'autre chose qu'une facture et uniquement via un transfert bancaire. Comptez vous utiliser autre chose que les cartes de crédit comme moyen de paiement ?**

***Jevgueni*** : En fait, même si ce n'est pas écrit explicitement sur notre site Web, la plupart des commandes importantes viennent avec une facture et un virement direct. Les sociétés nous contactent simplement par email et nous

nous arrangeons pour répondre à ce besoin. J'adorerais pouvoir faire cela d'une manière automatisée, mais je ne pense pas que notre banque prenne en charge l'un de ces services.

***David*** : Ce genre de choses est finalement assez simple à mettre en place. Envoyez simplement un email à sales@zeroturnaround.com.

## 3. L'outil JRebel

**Comment et quand le projet JRebel (anciennement JavaRebel) a-t-il vu le jour ?**

***Jevgueni*** : JRebel a démarré avec une idée courant août 2006. A cette époque, je travaillais sur le rechargement à chaud chez Aranea et j'ai soudain eu l'idée de faire du rechargement à chaud pour n'importe quel fragment de code Java. C'était assez compliqué, mais après quelques recherches j'ai pu mettre au point un prototype et le faire marcher en mars 2007. En septembre 2007 nous lancions une version publique en bêta et les premières ventes ont démarré en octobre la même année. Cela a été un parcours passionnant depuis.

***David*** : Ce vendredi 16 avril (2010) nous avons sorti JRebel 3.0 et nous mesurons le long chemin parcouru depuis la première version. JRebel se concentre toujours et encore sur l'élimination de la phase de redéploiement dans le développement Java EE, mais possède également un support pour les EDI, gère les changements sur la structure de classe, prend en charge le build instantané (avec des EAR, WAR, Maven, Ant), les changements dans la configuration (Spring, Guice, Struts 1 et 2, Log4J, Tapestry4, Wicket, Stripes, Velocity), et couvre les versions 1.6 à 1.4 de Java.

**Estimez-vous aujourd'hui avoir des concurrents sur le créneau couvert par JRebel ?**

***Jevgueni*** : Aucun qui ne nous inquiète.

***David*** : Cela sonne comme une réponse typique de commercial ! Comme nous nous amusons à inverser les rôles, je vais compléter la réponse pour celle-ci. La version courte est "oui, nous avons des concurrents". Il y a HotSwap, "Hot Redeploy" - qui se décline selon différents noms (comme "FastSwap") selon le conteneur qui l'implémente. Malheureusement, à la fois HotSwap et Hot Redeploy ont d'importantes lacunes... il est plus simple d'observer ce comparatif (Lien11) pour se faire une idée.

**Avec quel EDI JRebel est-il actuellement le mieux intégré ? ZeroTurnaround se charge-t-il de proposer une intégration pour tous les EDI principaux du marché ou s'appuie-t-il sur les communautés de ces derniers ?**

***Jevgueni*** : Nous fournissons un niveau de support environ équivalent pour Eclipse et IDEA. Le plugin NetBeans ne dispose pas de toutes les fonctionnalités que nous proposons pour les autres EDI. Il y a quelques contributions par la communauté (par exemple, le plugin IDEA était initialement une contribution), mais le plus souvent nous prenons en charge leur développement.

**Quels sont les prochains challenges pour JRebel après la version 3.0 ?**

***Jevgueni*** : Eh bien, il y a de nombreuses choses que nous avons initié avec la version 3.0 que nous allons continuer de complémenter. Le support complet de JPA en fait partie. Mais nous allons également mettre notre énergie sur le développement de LiveRebel, lequel dispose d'un gros potentiel et est attendu par plus de 200 sociétés. Cela se traduira aussi par une innovation importante pour JRebel.

**Quels sont les plugins que vous comptez rajouter par la suite à JRebel, pour le support d'autres frameworks. Je pense par exemple actuellement à JSF, dont la configuration n'est pas mise à jour par JRebel et pour lequel l'ajout de bean nécessite quand même un redémarrage, ou commons-chain, qui charge une seule fois ses catalogues.**

***Jevgueni*** : JSF est pris en charge dans la version 3.0, :) Mojarra est d'ores et déjà supporté et MyFaces le sera également bientôt. Nous aimerions désormais moins nous concentrer sur le support des frameworks et davantage à permettre à la communauté de contribuer aux plugins plus facilement. Il est déjà assez simple de créer des plugins pour JRebel, mais nous prévoyons de le rendre très vite encore plus simple. Si vous vous intéressez à l'écriture d'un plugin pour un framework en particulier, faites-moi signe directement par mail sur jevgeni@zeroturnaround.com

**Si ce n'est pas un secret, combien de licences, approximativement, avez-vous vendues de la version 2.2.1, quelle évolution envisagez-vous pour les ventes de la version 3 ?**

***Jevgueni*** : Pour être honnête, je ne suis pas sûr du nombre de têtes. C'est de l'ordre du millier. Je sais que nous avons plus de 10 000 téléchargements par mois, et cela augmente à la manière de la forme d'une crosse de hockey..., hmm... peut-être n'est ce qu'une expression valable au Canada ? Avec la version 3.0, nous modifions légèrement le modèle des licences, mais le changement le plus notable est l'ajout de l'Enterprise Add-On. Nous ajoutons le support de technologies anciennes qui donnent mal à la tête à bon nombre de gens... versions majeures de serveurs d'applications en fin de vie, EJB 1.x et 2.x, et l'ajout d'un serveur de licences qui permettra aux plus importants consommateurs de gérer leur parc de licences d'une manière plus efficace. Toutes les licences continueront à être sur une souscription annuelle et continueront à vous faire économiser en moyenne cinq semaines de temps de développement en redémarrage et redéploiement. Vraiment impressionnant.

## 4. La JVM et le système de gestion dynamique des modules

**Quelles limitations technologiques de la JVM aimeriez vous voir disparaître pour améliorer votre produit. Autrement dit, avez vous des fonctionnalités que vous aimeriez voir dans JRebel mais qu'il est techniquement impossible de réaliser à cause de la JVM ?**

***Jevgueni*** : En haut de ma liste : Interface Injection. Comparativement, cela devrait être simple à prendre en charge, étant donné que je suppose que cela ne nécessite pas nécessairement la mise à jour des vtables (qui sont inline dans la JVM de Sun et sont la raison pour laquelle HotSwap ne permet pas l'ajout de méthodes).

**ZeroTurnaround pourrait-il être intéressé de participer à l'amélioration dans OpenJDK du fonctionnement du ClassLoader et de HotSwap ?**

***Jevgueni*** : Ce serait certainement intéressant de faire cela, nous avons discuté de cette opportunité mais nous n'avons rien décidé pour le moment.

**Que pensez-vous de l'effet de mode OSGi ?**

***Jevgueni*** : C'est effectivement tendance. OSGi n'est utile qu'aux développeurs de frameworks et de serveurs, les applications ne devraient pas à avoir à s'en préoccuper. C'est une abstraction bas niveau utile, mais trop générique pour les applications. La bonne façon de l'utiliser consiste à construire une couche modulaire spécialisée par dessus.

J'ai également un problème avec la façon dont les class loaders sont utilisés. Il est bien connu que les dépendances circulaires avec OSGi peuvent occasionner des interblocages (deadlocks) de class loader, et je suis assez convaincu qu'OSGi favorise les leaks de class loader (ce n'est pas vraiment de sa faute, mais le modèle du class loader en Java n'a jamais été pensé pour être utilisé avec un graphe).

## 5. A venir...

**Espérez vous voir se développer une communauté autour de votre système de plugin, où tout un chacun hébergerait, peut être chez vous, ses propres plugins à destination du public ?**

***Jevgueni*** : Oui, c'est l'objectif. :) Nous envisageons quelque chose de ce genre, mais nous avons constaté que sans notre attention, les pages communautaires tendent à rapidement stagner. Maintenant que l'adoption de JRebel progresse de manière exponentielle, nous avons vraiment besoin de relancer cette communauté de plugins rapidement.

**LiveRebel, actuellement en bêta privée, est-il la suite logique de JRebel ? Pouvez-vous nous parler un peu de ce nouveau produit ?**

***David*** : Nous avons eu énormément de demandes de nos clients pour transposer la technologie JRebel du domaine du développement vers celui de la production et des applications en production. Cela a du sens pour nous d'étudier ce besoin de manière très sérieuse et de voir ce que nous pouvons proposer. J'ai relevé une citation quelque part qui disait que si Amazon.com venait à tomber, cela leur coûterait 31 000 $ par minute, et j'ai connaissance d'entreprises pour lesquels le coût serait plus élevé, donc nous aimerions aider les entreprises à conserver leurs applications en ligne même dans le cadre de procédures de mises à jour mineures. Pour ceux qui sont intéressés, nous avons un groupe pour la bêta privée de LiveRebel, auquel nous allons porter toute notre attention sur l'année à venir : Lien12.

**Pourrait-on voir apparaître dans l'offre de ZeroTurnaround des JRebel like pour d'autres plateformes, comme par exemple DotNET ?**

***Jevgueni*** : J'aimerais pouvoir répondre oui à cette question. :) Nous avons étudié la plateforme .NET et il lui manque les éléments nécessaires à la mise à disposition de fonctionnalités que nous avons pour Java. Cela est principalement dû à des manques dans le classloader qui est une formidablement puissante abstraction qui peut devenir dangereuse lorsqu'on en abuse.

## 6. Conclusion

**Un message à faire passer à la communauté francophone Java ?**

***David*** : Bien sûr allez vous impliquer dans vos JUG (Java User Groups) locaux. Au cours de ces dernières années, les communautés françaises de JUG ont réellement percé dans de nombreuses villes de France. C'est formidable de voir tant de gens se réunir pour parler de Java et s'aider mutuellement !

### 6.2. Liens

- Version originale de l'interview : Lien13.
- ZeroTurnaround : Lien14.
- Téléchargez la dernière version de JRebel : Lien15.
- Partagez vos retours sur JRebel : Lien16.
- Vos questions sur les outils pour Java : Lien17.

*Retrouvez l'interview d'Eric Siber et David Delbecq en ligne : Lien18*

## Personnaliser une ListView

Cet article est une introduction à la manipulation des listes sous Android via le composant ListView. Il permet de comprendre les bases de l'affichage des items.

### 1. Présentation

Sous Android, le composant principal pour afficher et gérer une liste de données est le composant ListView. Par défaut ce composant affiche une liste de chaînes de caractères. Selon les besoins, il peut être nécessaire d'afficher plus d'informations sur chaque ligne de la liste, ou même de venir effectuer des opérations particulières suite à des actions sur les lignes.

Dans un premier temps nous allons voir comment afficher une liste toute simple ne contenant que des chaînes de caractères, puis nous verrons comment personnaliser l'affichage des items de la liste et enfin comment ajouter des actions sur les items.

### 2. Liste de chaînes de caractères

Le SDK nous propose un type d'Activity particulier pour les vues affichant une liste : ListActivity (Lien19). Cette activity intègre de base pas mal de mécanismes pour gérer les listes.

Il y a également moyen de gérer tout depuis le début et d'afficher une liste depuis une Activity classique.

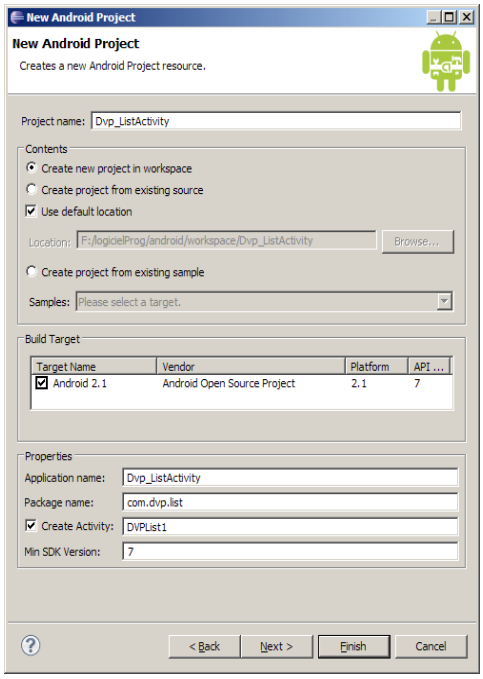
#### 2.1. Affichage avec une ListActivity

Dans un premier temps, créez un nouveau projet ("Dvp_ListActivity" par exemple). Choisissez le SDK que vous souhaitez ; pour ma part, je me baserais sur la dernière version (2.1), mais le code est également valide pour les versions antérieures.

Sur le wizard de création de projet, il vous est proposé de créer également une Activity, on va en profiter pour le faire tout de suite.

Choisissez comme nom pour votre Activity : "DVPList1". Le layout associé sera "main.xml".

Voici ce que vous devriez avoir sur votre wizard de création de projet :




Maintenant on va changer le type de notre DVPList1. À l'heure actuelle, c'est une Activity standard. Nous allons changer ça et la remplacer par une ListActivity.

```
public class DVPList1 extends Activity {
```

devient :

```
public class DVPList1 extends ListActivity {
```

Maintenant, notre activité sait gérer naturellement une liste de données. Le problème c'est qu'on n'en a pas encore créé dans notre layout. Pour ce faire, on ouvre le fichier main.xml et on ajoute le composant ListView.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

devient :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
        <ListView android:id="@android:id/list"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
        </ListView>
</LinearLayout>
```

Une chose importante à ce niveau-là est l'identifiant du composant ListView : "@android:id/list". Ici on ne peut pas mettre un identifiant personnel comme on peut le faire pour les autres composants, il faut mettre celui indiqué pour que les mécanismes de gestion de vues (ceux présents dans ListActivity) arrivent à bien fonctionner. En effet, en mettant un identifiant personnalisé, l'objet ListActivity n'arriverait pas à retrouver notre composant ListView dans notre layout et on obtiendrait des exceptions du style :

```
Caused by: java.lang.RuntimeException: Your
content must have a ListView whose id attribute
is 'android.R.id.list'
```

Maintenant que notre layout contient une liste et que notre activité sait la gérer, on peut donc remplir celle-ci. Pour ce faire, nous allons tout d'abord initialiser un tableau de chaînes de caractères de la manière suivante :

```
private String[] mStrings = {
            "AAAAAAAA", "BBBBBBBB", "CCCCCCCC",
"DDDDDDDD", "EEEEEEEE",
            "FFFFFFFF", "GGGGGGGG", "HHHHHHHH",
"IIIIIIII", "JJJJJJJJ",
            "KKKKKKKK", "LLLLLLLL", "MMMMMMMM",
"NNNNNNNN", "OOOOOOOO",
            "PPPPPPPP", "QQQQQQQQ", "RRRRRRRR",
"SSSSSSSS", "TTTTTTTT",
            "UUUUUUUU", "VVVVVVVV", "WWWWWWWW",
"XXXXXXXX", "YYYYYYYY",
            "ZZZZZZZZ"
};
```

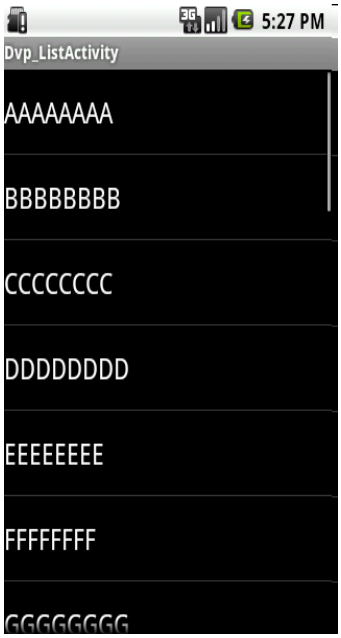
Ici l'initialisation du tableau est statique, mais on peut très bien imaginer que les informations contenues dans ce tableau proviennent par exemple d'un webservice, d'un fichier XML, d'un provider du système, etc.

Il ne nous reste plus qu'à afficher ces données dans notre liste. Pour ce faire, nous allons utiliser les fonctionnalités proposées par ListActivity :

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, mStrings);

    setListAdapter(adapter);
}
```

L'adapter permet de gérer les données et de réaliser le mapping relationnel entre les données et l'IHM. Le premier paramètre (this) permet d'identifier le contexte dans lequel notre adapter va fonctionner. Le deuxième paramètre "android.R.layout.simple_list_item_1" permet d'indiquer la présentation utilisée pour les items de notre liste. Ici, il s'agit d'une présentation simple pour les chaînes de caractères incluses dans le SDK.

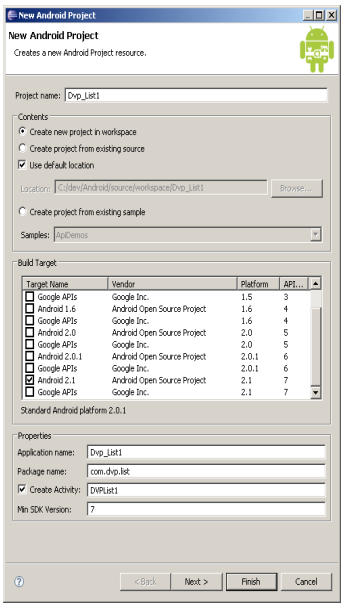
L'appel de cette méthode permet donc d'afficher notre liste. Ainsi, si nous exécutons notre projet, nous obtenons ceci :




Les sources de cette application sont téléchargeable ici : Lien20.

## 2.2. Affichage sans ListActivity

Maintenant que nous avons vu comment bénéficier des avantages des ListActivity, nous allons voir comment reproduire ce mécanisme de gestion de listes de A à Z. Pour ce faire, nous allons repartir de zéro et récréer un nouveau projet comme indiqué sur l'image suivante :



Maintenant nous allons éditer le fichier "main.xml" afin de lui rajouter un composant ListView.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

devient :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
        android:orientation="vertical"
android:layout_width="fill_parent"
        android:layout_height="fill_parent">

        <ListView android:id="@+id/ListView01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
        </ListView>
</LinearLayout>
```

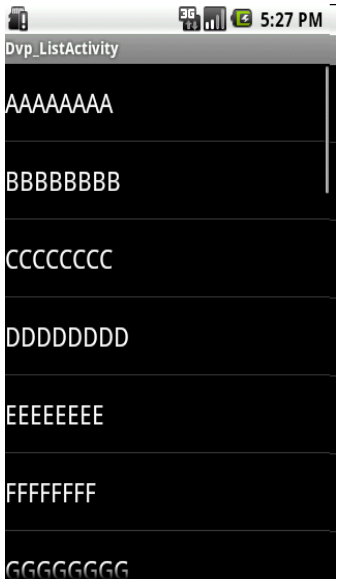
Dans notre Activity DVPList1, de la même façon que dans l'exemple précédent, nous allons initialiser un tableau de String de manière statique (à titre d'exemple) :

```
private String[] mStrings = {
            "AAAAAAAA", "BBBBBBBB", "CCCCCCCC",
"DDDDDDDD", "EEEEEEEE",
            "FFFFFFFF", "GGGGGGGG", "HHHHHHHH",
"IIIIIIII", "JJJJJJJJ",
            "KKKKKKKK", "LLLLLLLL", "MMMMMMMM",
"NNNNNNNN", "OOOOOOOO",
            "PPPPPPPP", "QQQQQQQQ", "RRRRRRRR",
"SSSSSSSS", "TTTTTTTT",
            "UUUUUUUU", "VVVVVVVV", "WWWWWWWW",
"XXXXXXXX", "YYYYYYYY",
            "ZZZZZZZZ"
};
```

Maintenant, nous allons créer un adapter (comme dans l'exemple précédent) afin de fournir notre liste de données au composant ListView :

```
//Création de l'adapter
ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, mStrings);

//Récupération du ListView présent dans notre IHM
ListView list =
(ListView)findViewById(R.id.ListView01);

//On passe nos données au composant ListView
list.setAdapter(adapter);
```

Voici ce que nous obtenons à l'exécution :



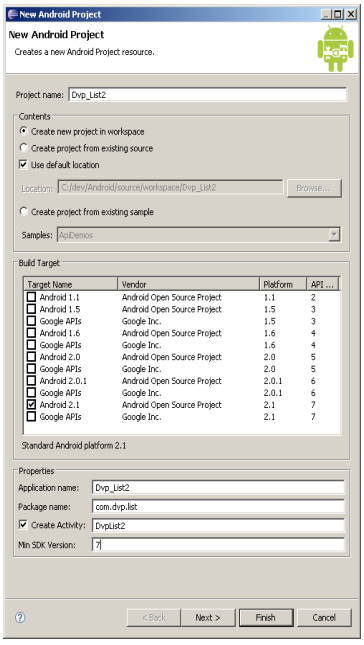

Comme vous pouvez le voir, il n'y a pas de différence notable entre les deux versions, le rendu est le même. Pour l'instant le seul avantage d'utiliser une ListActivity par rapport à un Activity classique réside dans le fait qu'il n'y a pas besoin de récupérer l'instance du composant ListView dans l'IHM.

Les sources de cette application sont téléchargeable ici : Lien21.

## 3. Personnalisation des items

Dans cette partie, nous allons voir comment personnaliser notre liste et surtout, comment personnaliser la présentation des items de la liste. Le thème de cet exercice sera d'afficher une liste de personnes. Une personne aura un nom, un prénom, un genre (Masculin / Féminin).

Dans un premier temps, nous allons créer un projet "DVP_List2", tel que décrit sur ce wizard :




Ensuite, nous allons créer une classe pour représenter les personnes. Cette classe contient trois propriétés : "nom",

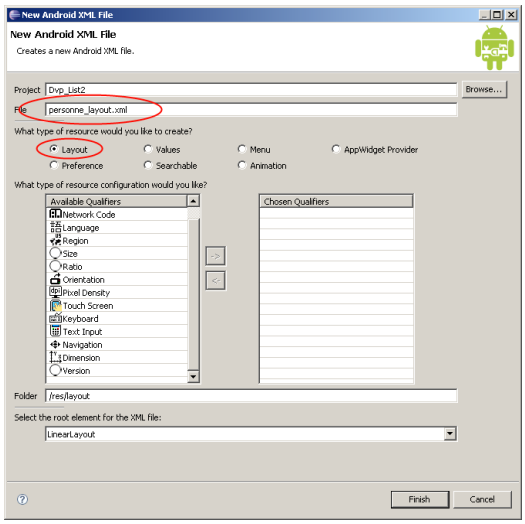
"prenom" et "genre". Afin de simplifier les développements et l'exemple, on rajoute une méthode static dans cette classe pour remplir et renvoyer une liste de personnes. Le code de la classe donne donc :

```
package com.dvp.list;

import java.util.ArrayList;

public class Personne {
	public final static int MASCULIN = 1;
	public final static int FEMININ = 2;

	public String nom;
	public String prenom;
	public int genre;

	public Personne(String aNom, String
aPrenom, int aGenre) {
		nom = aNom;
		prenom = aPrenom;
		genre = aGenre;
	}

	/**
	 * Initialise une liste de personnes
	 * @return une liste de "Personne"
	 */
	public static ArrayList<Personne>
getAListOfPersonne() {
		ArrayList<Personne> listPers =
new ArrayList<Personne>();

		listPers.add(new Personne("Nom1",
"Prenom1", FEMININ));
		listPers.add(new Personne("Nom2",
"Prenom2", MASCULIN));
		listPers.add(new Personne("Nom3",
"Prenom3", MASCULIN));
		listPers.add(new Personne("Nom4",
"Prenom4", FEMININ));
		listPers.add(new Personne("Nom5",
"Prenom5", FEMININ));
		listPers.add(new Personne("Nom6",
"Prenom6", MASCULIN));
		listPers.add(new Personne("Nom7",
"Prenom7", FEMININ));
		listPers.add(new Personne("Nom8",
"Prenom8", MASCULIN));
		listPers.add(new Personne("Nom9",
"Prenom9", MASCULIN));
		listPers.add(new
Personne("Nom10", "Prenom10", FEMININ));
		listPers.add(new
Personne("Nom11", "Prenom11", MASCULIN));
		listPers.add(new
Personne("Nom12", "Prenom12", MASCULIN));
		listPers.add(new
Personne("Nom13", "Prenom13", FEMININ));
		listPers.add(new
Personne("Nom14", "Prenom14", MASCULIN));

		return listPers;
	}
}
```

Maintenant, dans le layout XML (main.xml), rajoutez le composant ListView. Le code XML du layout doit correspondre à ceci :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
	android:orientation="vertical"
android:layout_width="fill_parent"
	android:layout_height="fill_parent">

	<ListView android:id="@+id/ListView01"
	    android:layout_width="wrap_content"
	    android:layout_height="wrap_content">
	</ListView>
</LinearLayout>
```

Les personnes possèdent trois propriétés. Il n'est donc plus possible de passer par le layout standard pour les afficher. Nous allons donc créer un nouveau layout et l'utiliser pour afficher les items de la liste. Ajoutez un nouveau fichier XML dans le répertoire layout de votre application. (Clic droit sur votre projet, choisissez "New" puis "Android XML file"). Configurez le wizard tel qu'indiqué sur l'image suivante :




Dans ce nouveau layout, nous allons donc ajouter deux TextView pour décrire les nom et prénom de chaque personne. Le genre de la personne sera décrit par la couleur de fond de notre item : bleu pour les garçons, rose pour les filles :). Le XML de notre item doit donc ressembler à ça :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
	android:layout_width="wrap_content"
android:layout_height="wrap_content"
	android:id="@+id/LL_Fond">

	<TextView android:text="Nom"
android:id="@+id/TV_Nom"
	    android:layout_width="wrap_content"
	    android:layout_height="wrap_content">
		</TextView>

	<TextView android:text="Prénom"
android:id="@+id/TV_Prenom"
	    android:layout_width="wrap_content"
	    android:layout_height="wrap_content">
		</TextView>
</LinearLayout>
```

Il ne nous reste plus qu'à afficher la liste des personnes en

utilisant notre layout. Pour ce faire, nous allons créer un objet qui se chargera de gérer le mapping entre nos données et le layout des items. Ce composant sera basé sur un Adapter. Dans Eclipse, créez une nouvelle classe : "PersonneAdapter". Faites-la hériter de "BaseAdapter".

```
public class PersonneAdapter extends BaseAdapter
{
```

L'Adapter va gérer une liste de personnes et va s'occuper également de l'affichage. On va donc lui ajouter trois propriétés

```
// Une liste de personnes
private List<Personne> mListP;

//Le contexte dans lequel est présent notre
adapter
private Context mContext;

//Un mécanisme pour gérer l'affichage graphique
depuis un layout XML
private LayoutInflater mInflater;
```

Le constructeur va alors ressembler à ceci :

```
public PersonneAdapter(Context context,
List<Personne> aListP) {
  mContext = context;
  mListP = aListP;
  mInflater = LayoutInflater.from(mContext);
}
```

Le LayoutInflater permet de parser un layout XML et de le transcoder en IHM Android. Pour respecter l'interface BaseAdapter, il nous faut spécifier la méthode "count()". Cette méthode permet de connaître le nombre d'items présents dans la liste. Dans notre cas, il faut donc renvoyer le nombre de personnes contenu dans "mListP".

```
public int getCount() {
  return mListP.size();
}
```

Ensuite, il y a deux méthodes pour identifier les items de la liste. Une pour connaître l'item situé à une certaine position et l'autre pour connaître l'identifiant d'un item en fonction de sa position.

```
public Object getItem(int position) {
  return mListP.get(position);
}

public long getItemId(int position) {
  return position;
}
```

Maintenant il faut surcharger la méthode pour renvoyer une "View" en fonction d'une position donnée. Cette view contiendra donc une occurrence du layout "personne_layout.xml" convenablement renseignée (avec le nom et prénom au bon endroit).

```
public View getView(int position, View
convertView, ViewGroup parent) {
  LinearLayout layoutItem;
  //(1) : Réutilisation des layouts
  if (convertView == null) {
      //Initialisation de notre item à partir
du  layout XML "personne_layout.xml"
    layoutItem = (LinearLayout)
mInflater.inflate(R.layout.personne_layout,
parent, false);
  } else {
      layoutItem = (LinearLayout) convertView;
  }

  //(2) : Récupération des TextView de notre
layout
  TextView tv_Nom =
(TextView)layoutItem.findViewById(R.id.TV_Nom);
  TextView tv_Prenom =
(TextView)layoutItem.findViewById(R.id.TV_Prenom)
;

  //(3) : Renseignement des valeurs
  tv_Nom.setText(mListP.get(position).nom);
  tv_Prenom.setText(mListP.get(position).prenom);

  //(4) Changement de la couleur du fond de notre
item
  if (mListP.get(position).genre ==
Personne.MASCULIN) {
    layoutItem.setBackgroundColor(Color.BLUE);
  } else {
      layoutItem.setBackgroundColor(Color.MAGEN
TA);
  }

  //On retourne l'item créé.
  return layoutItem;
}
```

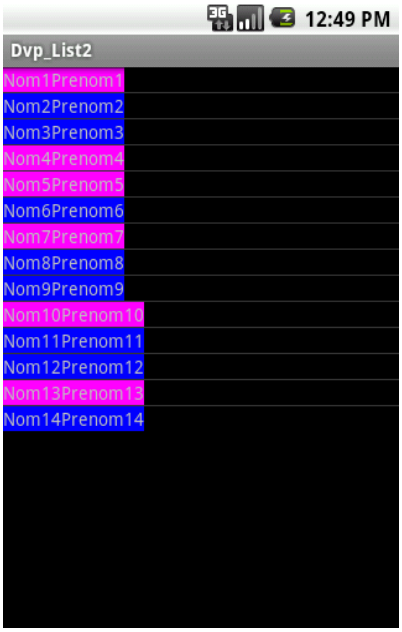
Le paramètre "convertView" permet de recycler les éléments de notre liste. En effet, l'opération pour convertir un layout XML en IHM standard est très coûteuse pour la plateforme Android. On nous propose ici de réutiliser des occurrences créées qui ne sont plus affichées. Donc si ce paramètre est "null" alors, il faut "inflater" notre layout XML, sinon on le réutilise (1). Maintenant que notre layout est initialisé, on peut récupérer les deux champs texte qui y sont présents (2) afin de modifier leur valeur (3). Afin de respecter notre contrat, les garçons doivent être en bleu et les filles en rose. Il nous suffit donc de changer la couleur du fond (4).

Pour résumer, nous avons une vue contenant une liste (main.xml), une description de la présentation d'un item de la liste (personne_layout.xml) et un composant pour gérer le mapping donnée - IHM (PersonneAdapter.java). Il ne nous reste donc plus qu'à afficher notre liste. Pour ce faire, modifiez l'Activity principale, DVPList2.java :

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    //Récupération de la liste des personnes
    ArrayList<Personne> listP =
Personne.getAListOfPersonne();

        //Création et initialisation de l'Adapter
pour les personnes
    PersonneAdapter adapter = new
PersonneAdapter(this, listP);
```

```
    //Récupération du composant ListView
    ListView list =
(ListView)findViewById(R.id.ListView01);

    //Initialisation de la liste avec les données
    list.setAdapter(adapter);
}
```

Maintenant, si vous exécutez votre programme, vous devriez obtenir ceci :




Les sources de cette application sont téléchargeables ici : Lien22.

## 4. Ajout d'évènements

Nous allons maintenant voir comment ajouter un mécanisme pour écouter ce qu'il se passe sur notre liste. Nous allons simplement enrichir l'application précédente. Le but de ce prochain exemple sera d'écouter si l'utilisateur clique sur le nom d'une personne.

Dans un premier temps, créez une interface pour votre listener personnalisé et dans votre fichier "PersonneAdapter.java", rajoutez la partie suivante :

```
/**
 * Interface pour écouter les évènements sur le
nom d'une personne
 */
public interface PersonneAdapterListener {
    public void onClickNom(Personne item, int
position);
}
```

Ensuite, créez le mécanisme pour gérer l'ajout de listener sur notre adapter :

```
//Contient la liste des listeners
private ArrayList<PersonneAdapterListener>
mListListener = new
ArrayList<PersonneAdapterListener>();
/**
 * Pour ajouter un listener sur notre adapter
 */
public void addListener(PersonneAdapterListener
aListener) {
    mListListener.add(aListener);
}
```

La prochaine méthode permet de prévenir tous les listeners en même temps pour diffuser une information :

```
private void sendListener(Personne item, int
position) {
    for(int i = mListListener.size()-1; i >= 0;
i--) {
        mListListener.get(i).onClickNom(item,
position);
    }
}
```

Le mécanisme pour ajouter / prévenir les listeners est en place. On peut ajouter le listener interne sur le clic du TextView du nom des personnes. Pour ce faire, dans la méthode "getView" de l'adapter, ajoutez ceci :

```
if (mListP.get(position).genre ==
Personne.MASCULIN) {
    layoutItem.setBackgroundColor(Color.BLUE);
} else {
    layoutItem.setBackgroundColor(Color.MAGENTA);
}

//------------ Début de l'ajout -------
//On mémorise la position de la "Personne" dans
le composant textview
tv_Nom.setTag(position);
//On ajoute un listener
tv_Nom.setOnClickListener(new OnClickListener() {

        @Override
        public void onClick(View v) {
                //Lorsque l'on clique sur le nom,
on récupère la position de la "Personne"
                Integer position =
(Integer)v.getTag();

                //On prévient les listeners qu'il
y a eu un clic sur le TextView "TV_Nom".
                sendListener(mListP.get(position)
, position);

        }

});
//------------ Fin de l'ajout -------

return layoutItem;
```

Maintenant, nous allons retravailler sur notre activity principale (DVPList2.java) pour qu'elle écoute les évènements sur notre liste. Tout d'abord, faites-là implémenter notre interface "PersonneAdapterListener" :

```
public class DvpList2 extends Activity implements
PersonneAdapterListener {
```

Rajoutez la méthode déclenchée lors d'un clic sur le nom d'une personne. Dans notre cas, un popup s'ouvrira pour donner le nom de la personne cliqué.

```
public void onClickNom(Personne item, int
position) {
```

```
        Builder builder = new
AlertDialog.Builder(this);
        builder.setTitle("Personne");

        builder.setMessage("Vous avez cliqué
sur : " + item.nom);
        builder.setPositiveButton("Oui", null);
        builder.setNegativeButton("Non", null);
        builder.show();
}
```

Et enfin, branchez votre listener sur votre liste :

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    ArrayList<Personne> listP =
Personne.getAListOfPersonne();
    PersonneAdapter adapter = new
PersonneAdapter(this, listP);

    //Ecoute des évènements sur votre liste
    adapter.addListener(this);

    ListView list =
(ListView)findViewById(R.id.ListView01);

    list.setAdapter(adapter);
}
```

Maintenant, si vous exécutez votre programme, vous devriez obtenir ceci :


Les sources de cette application sont téléchargeables ici : Lien23.
Je tiens également à vous faire remarquer que si vous souhaitez écouter les évènements (onClick) sur une liste depuis une activity de type "ListActivity", il vous suffit de surcharger la méthode onListItemClick :

```
protected void onListItemClick(ListView l, View
v, int position, long id)
```

A partir de là, vous pourrez récupérer l'item qui a été sélectionné par l'utilisateur. Voici un exemple d'implémentation qui peut être inséré dans la classe DVP_List1.java de l'exemple du paragraphe I :

```
@Override
protected void onListItemClick(ListView l, View
v, int position, long id) {
  super.onListItemClick(l, v, position, id);

  Builder builder = new
AlertDialog.Builder(this);
  builder.setTitle("Item");

  builder.setMessage("Vous avez cliqué sur : " +
mStrings[position]);
  builder.setPositiveButton("Oui", null);
  builder.setNegativeButton("Non", null);
  builder.show(); }
```

## 5. Conclusion

Pour conclure, l'utilisation des composants pour lister les données sous Android peut dans un premier temps paraître très simple pour une utilisation basique, mais offre également de puissants moyens de personnalisation dès que l'on souhaite enrichir un peu plus notre interface. Ce tutoriel vous aura permis de découvrir les bases de ces personnalisations, il y en a encore bien d'autres, notamment les optimisations d'affichage (en plus de celle que l'on a découvert dans le paragraphe III).

## 6. Liens

Site du développeur : Lien24
FAQ Android : Lien25
Installation de l'environnement : Lien26

*Retrouvez l'article de Mickaël Le Trocquer en ligne : Lien27*

## Compilation détaillée de PHP sous Linux

PHP est écrit en C et, à ce titre, il est compilable en langage machine. Nous allons détailler comment fonctionne ce processus sous Linux, ainsi qu'une partie de l'écosystème de PHP : ses extensions, les bibliothèques utilisées, son moteur... Pour suivre cet article, vous devez connaître le langage PHP et avoir quelques notions d'UNIX, c'est tout. Nous effleurerons également quelques concepts relatifs au langage C, sans rentrer dans les détails. La version de PHP considérée est 5.3.x.

### 1. Premier pas avec la compilation de PHP

Avant tout, compiler c'est quoi ? C'est très simple : c'est transformer un langage informatique en un autre. Le moteur du langage PHP est écrit en C, pour obtenir l'environnement PHP il faut donc le compiler, c'est-à-dire transformer son code source C en un code binaire exécutable par le processeur de la machine cible. Ce code sera encapsulé dans un binaire que vous pourrez exécuter : ce sera PHP !
PHP est open source, vous pouvez donc télécharger son code source, l'étudier, le modifier et le transformer (le compiler).

#### 1.1. Quelques questions pour un bon départ

*Je peux déjà récupérer PHP sur le site officiel, sans télécharger ses sources* ? Oui, vous pouvez télécharger ce que l'on appelle "les binaires". En fait, il s'agit d'un PHP compilé par les développeurs du langage. Il a été compilé "basiquement", pour répondre à une très large palette de besoins, mais pas forcément aux vôtres.

*Je peux aussi trouver PHP dans des packages comme WampServer, Xamp ou ZendServer, des précisions ?* Oui ; ces logiciels sont à l'opposé du "pourquoi compiler moi-même ?", ils encapsulent un PHP sur lequel vous n'avez la plupart du temps aucun pouvoir de décision. Il est aussi possible qu'ils aient eux-mêmes compilé le PHP qu'ils embarquent avec des options qui en restreignent l'usage (par exemple, il n'est pas possible de charger telle extension donnée). Ce sont des solutions convenables si vous ne souhaitez vraiment pas vous embêter, mais soyez conscients que vous laissez des personnes tierces configurer PHP du cœur à l'épiderme, sans que vous ayez la main sur quoi que ce soit.

*Pourquoi compiler PHP moi-même alors ?* Il y a beaucoup d'avantages à cela : déjà vous pourrez utiliser des options de compilation propres à votre plateforme. Vous obtiendrez ainsi un PHP prêt à être exécuté sur votre machine, pour son processeur, avec toutes les optimisations qui le concernent. Par rapport aux binaires PHP que vous pouvez télécharger pour votre système, vous gagnerez en performance et vous pourrez atteindre (avec des connaissances et de l'expérience) un gain pouvant aller jusqu'à 10 ou 15 % dans certains cas.
Ensuite, compiler soi-même PHP c'est pouvoir le personnaliser à la source : inclure ou exclure des extensions au langage (et vous savez qu'il en possède des nombreuses pas forcément disponibles sans passer par la case compilation manuelle), ou encore activer des options très spéciales dont nous reparlerons plus tard.
Enfin, cela vous permettra sans aucun doute de mieux maîtriser le langage : la compilation peut être assimilée à la fabrication et la naissance d'un PHP faites par vos soins, de vos propres mains.

*Pourquoi faire cela sous Linux ?* PHP a été conçu pour Unix, puis a été porté sous Windows et autres systèmes non Unix. Quoi qu'on en dise, PHP est plus performant sous Linux et beaucoup plus facile à compiler sous ce système (au sein duquel il est né il y a maintenant 15 ans) que sous un autre, particulièrement Windows. Lorsque PHP est sous Linux, considérez cela comme un poisson dans l'eau : il est dans son milieu, dispose de tout ce dont il a besoin nativement et se sent en pleine forme.

*Compiler PHP sous Linux, c'est difficile ?* J'espère que ce tutoriel saura vous démontrer que NON : absolument pas, c'est un jeu d'enfant. Vous devez simplement être familier de Linux (sans pour autant être un pro), c'est tout.

#### 1.2. Préparation du système

Nous allons utiliser un système Ubuntu Server (Lien28). Historiquement, le système le moins "galère" pour compiler PHP et l'utiliser est Debian.
Ubuntu est une branche très proche de Debian, les manipulations notées ici seront quasi identiques sous Debian. Concernant les autres systèmes Linux, ça ne change pas beaucoup, c'est vraiment très ressemblant. Idem pour les systèmes Unix, il existe quelques différences avec Linux selon les distributions, mais vraiment rien de très méchant.

Utilisez une machine virtuelle ! Pour installer un système Ubuntu Server sans toucher à votre machine, utilisez une machine virtuelle. Si votre système hôte est Windows, vous vous retrouverez avec un poste de développement PHP sous Windows, mais pouvant exécuter un PHP totalement personnalisé et créé par vos soins sous Linux, ce qui offre des avantages considérables.
Je vous conseille l'utilisation de VirtualBox (Lien29) comme hyperviseur, si vous ne savez pas vers quoi vous tourner.

A partir de maintenant, nous supposons que vous disposez d'un système Ubuntu Server, classiquement installé sans

option superflue. Nous supposons aussi que le système est à jour. Exécutez les commandes *aptitude update* puis *aptitude safe-upgrade* si ce n'est pas le cas.

## 2. Au cœur de PHP

Il ne va pas s'agir ici de décrire précisément le fonctionnement du langage, mais plutôt de manière globale. C'est essentiel pour pouvoir comprendre la compilation des sources.
Vous savez que PHP est modulaire, entendons par là qu'il dispose d'un corps (des fonctionnalités de base) enrichi par des extensions. Que ce soient les extensions de PHP ou bien son cœur le plus profond, tout son code source est écrit en C (à 98 %). Or en C, comme en PHP, on se sert d'extensions, que l'on appelle des bibliothèques ("libraries" en anglais).

Les bibliothèques C dont se sert PHP peuvent être vues comme un ensemble de fonctions C dont le code source de PHP tire parti.
Prenons un exemple : les fonctionnalités XML de PHP. Vous connaissez DOM, SimpleXml, SOAP etc. Ces fonctionnalités ont été en partie codées par les développeurs de PHP. En partie seulement, car analyser du code XML, trouver les noeuds, valider avec des DTD ou des schémas... sont autant de fonctionnalités qui existent aussi hors de PHP. Prenez un langage comme Python : il sait aussi faire tout cela.
Ainsi, les fonctionnalités communes sont écrites dans des bibliothèques C qu'on appelle **bibliothèques partagées**, ou **shared libraries**. Notez au passage que lorsque ces bibliothèques sont compilées, on appelle cela des **shared objects**, dont l'extension est **.so** , je pense que ça vous dit quelque chose non ?

Pour résumer : le code source de PHP repose sur tout un tas de bibliothèques C dont vous aurez besoin pour compiler PHP lui-même. Si une bibliothèque manque... Eh bien en PHP il se passe quoi ? En général : **fatal error**, il manque du code source et ce sera pareil pour compiler PHP, s'il manque une bibliothèque utilisée par le code source que vous tentez de compiler, la compilation échouera fatalement.

### 2.1. Dépendances des bibliothèques C

Nous n'allons pas entrer dans les détails de la programmation C, ni même de la compilation de code C. Des articles et des tonnes de livres existent à ce sujet.
Côté pratique, les questions suivantes ressortent donc : de quelles bibliothèques C PHP a-t-il besoin pour être compilé ? Où trouver ces bibliothèques ?

Heureusement, sous Linux tout a été prévu. Avant de lancer une compilation, nous allons lancer un script qui va permettre deux choses : préciser quelles sont les options, les briques, les extensions que l'on veut compiler dans PHP, et surtout : est-ce que celles-ci sont disponibles sur le système actuel ?
On appelle cela *le script configure*.
Un nombre incalculable de projets sous Linux utilise ce système-là qui date d'il y a très longtemps. Le noyau Linux lui-même utilise un tel système de compilation ainsi que des bibliothèques C utilisées par PHP.

Lorsqu'une bibliothèque dépendante va manquer, le *script configure* échouera avec un message d'erreur. Ce message est peu explicite pour un néophyte, mais pour un informaticien avec un peu de jugeote, il est suffisamment clair.
C'est là que le système d'exploitation entre en jeu : soit celui-ci est bourré à craquer de bibliothèques C, auquel cas il est fort peu probable que le *script configure* échoue, soit il est raisonnablement lourd en bibliothèques C, auquel cas il faut qu'il dispose d'un gestionnaire de paquets suffisamment riche pour pouvoir récupérer celles-ci facilement.
Ubuntu Server ne dispose pas de tout ce qu'il faut nativement pour compiler PHP sur le tas (presque), mais dispose de dépôts logiciels riches et on trouvera toutes les bibliothèques que l'on souhaite lui ajouter assez rapidement.
Bien sûr il reste aussi la possibilité de télécharger la bibliothèque directement sur Internet sur le site de celle-ci, c'est plus pénible mais parfois nécessaire si le gestionnaire de paquets ne connaît pas la bibliothèque en question ou ne possède pas la bonne version.

### 2.2. L'architecture de PHP (simplifiée)

La compilation, c'est aussi la possibilité d'ajouter de nombreuses briques au mur PHP que vous tentez de monter (schématiquement). Au travers du *script configure*, vous allez pouvoir ajouter, ou au contraire retirer des fonctionnalités au langage PHP. Très souvent ce seront des extensions (je suppose que tout le monde connait le système d'extension de PHP), parfois ce seront des options et vous allez voir que PHP est très riche sur ce point.

PHP se décompose globalement en trois parties : le moteur, le cœur et les extensions.
Sans rentrer dans les détails, le moteur est la partie la plus indispensable et la plus complexe : le ZendEngine.
Là-dessus se greffe le cœur : un ensemble de fonctionnalités vitales à PHP (les fonctions de base, comme fopen() - les tableaux - les chaines), c'est son enveloppe minimale.
Enfin viennent les extensions qui enrichissent le langage.

Lors de la compilation, vous pouvez ne choisir que le cœur (avec le moteur bien sûr) : vous aurez alors un PHP extrêmement léger, très rapide, mais qui ne saura en contrepartie pas faire grand chose.
Au contraire, vous aurez aussi la possibilité de compiler de nombreuses extensions dans PHP : le langage deviendra alors extrêmement riche en fonctionnalités, mais au prix de performances légèrement moindres et d'un binaire PHP compilé très lourd (plusieurs dizaines de Mo). A vous de voir, c'est vous le capitaine à bord du navire !

Le *script configure* utilise une configuration par défaut que vous trouvez sur php.net. Par exemple, sans rien personnaliser, PHP est compilé avec le support complet de XML.
Plus vous ajouterez d'extensions ou de possibilités au langage, plus il vous faudra de bibliothèques C tierces, et plus le temps de compilation sera long. Aussi, il est tout à fait possible à la compilation d'embarquer des extensions PECL dans PHP. PECL (PHP Extensions Code Libraries (Lien30)) est un dépôt d'extensions qui n'ont pas été

choisies pour faire partie du code source distribué de PHP, ni des binaires PHP par défaut. Mais PECL regorge d'extensions dont vous ignorez à coup sûr l'existence, et qui risquent de vous étonner. Nous verrons cela par la suite.

Enfin, sachez qu'il existe deux modes de liaisons des extensions dans PHP : **statique** ou **partagée** (on dit aussi **dynamique**). On retombe là sur des concepts de C et du développement informatique en général. Retenez simplement que la compilation d'une extension de manière **statique** la fusionnera dans le cœur de PHP et celle-ci ne sera plus désactivable notamment au moyen de php.ini. Dans ce mode-là, l'extension fait partie de PHP, ce qui offre un gain en performance, surtout si vous utilisez (en PHP donc) l'extension de manière intensive.
En mode **partagé**, l'extension est chargée par un mécanisme spécifique, au lancement de PHP (ou via sa fonction dl()) : cela offre l'avantage de pouvoir démarrer PHP avec ou sans cette extension et donc d'alléger son empreinte mémoire et son poids au runtime (au démarrage du langage) ou encore d'utiliser alternativement plusieurs versions d'une même extension. En contrepartie, le chargement dynamique lie l'extension au cœur par "des ficelles" (schématiquement) : l'extension sera moins performante que si elle est liée statiquement, ça se joue à très peu mais si vous utilisez l'extension de manière intensive dans un environnement à haute charge, ce détail a son importance.

## 3. Préparation des sources

Après ces quelques notions simplifiées, nous pouvons démarrer. Avant de compiler, il faut récupérer les sources et les préparer au moyen du *script configure* livré avec les sources.

Nous supposons que l'installation se fait dans votre /home, de manière à ce que votre utilisateur possède tous les droits nécessaires sur son arborescence.
Une installation en conditions de production se ferait plutôt dans /usr/local, et nécessiterait des droits *root*.

préparation des sources
```
/> cd ~
~> mkdir -p monphp/src && cd monphp/src
~/monphp/src> wget http://fr.php.net/get/php-5.3.2.tar.gz/from/this/mirror
~/monphp/src> tar xzf php-5.3.2.tar.gz
~/monphp/src> cd php-5.3.2
```

Nous venons de créer une arborescence minimale sous /home/votre-utilisateur : un dossier de travail (monphp) puis un dossier contenant les sources (src). Nous avons téléchargé PHP5.3 dedans (la version la plus récente à la date d'écriture de cet article), nous l'avons décompressé et nous sommes dans son dossier.
Dès lors, nous pouvons lancer le *script configure* avec l'option *--help*

Le configure de PHP
```
~/monphp/src/php-5.3.2> ./configure --help

Voir le résultat complet de la commande en ligne (Lien31).
```

Bon, les habitués de Linux et de la compilation ne seront pas surpris (et sauront d'ailleurs continuer tout seuls).

Nous n'allons pas tout détailler, seulement certaines parties qui vous permettront par la suite de vous lancer et d'expérimenter.
Je vous conseille vivement - si vous découvrez un "nouveau terrain" en voyant le résultat de *configure* - de vous familiariser avec le langage C sous Linux. Vous découvrirez alors *configure* en détail, mais aussi l'outil *make*, les *autotools* Unix et tout le vaste univers qui tourne autour.

Les options sont séparées en sections, celles du dessus ne nous intéressent pas ici. La partie *SAPI modules* permet comme son nom l'indique de configurer les modules SAPI. Une Server Application Programming Interface (SAPI) est une interface d'interaction avec PHP, les plus connues sont CLI (ligne de commandes), APXS (Apache), CGI. L'utilisation de embed (embarqué) est aussi intéressante pour des programmes C qui veulent intégrer de la syntaxe PHP, nous y reviendrons.
Vient ensuite la partie *General settings* dont je vous laisse deviner l'utilité puis *Extensions* qui est la plus volumineuse et la plus utile : quoi embarquer dans PHP ?
On pourra noter à la fin les sections *PEAR*, *Zend* (configuration du moteur et de la machine virtuelle), *TSRM* (gestion de la Thread Safety), *Libtool* (Options générales de la LibTool Unix).

## 4. Installation basique

Toutes les options de configuration sont à passer en paramètres à *configure*.
**--prefix=/home/julien/monphp** permet d'indiquer où installer PHP une fois qu'il a été compilé. Par défaut, */usr/local* est utilisé, mais des droits *root* seront nécessaires pour écrire dans ce répertoire. Nous avons choisi d'installer dans */home/julien/monphp* que nous avons crée précédemment. Evidemment, vous aurez remplacé *julien* par votre nom de compte.

A ce stade, il est possible de lancer la commande. S'en suivra alors une préparation de PHP et une vérification de toutes les dépendances nécessaires.
Un détail important : tout ce qui est **--disable-xxx** permet de désactiver une fonctionnalité alors que **--enable-xxx** ou **--with-xxx** permet d'activer une fonctionnalité.
Vous en déduisez donc que tout ce qui vous propose **--disable-xxx** suppose que par défaut, la fonctionnalité est activée. On peut donc remarquer que, par exemple, la présence de **--disable-phar** indique que le support de Phar sera activé par défaut si on ne précise pas le contraire, et inversement : la présence de, par exemple, **--enable-soap** indique que par défaut le support de Soap n'est pas inclus.
De version en version de PHP, certaines options changent : les développeurs de PHP considèrent que pour telle version, l'option n'est pas activée par défaut, par contre elle demeure le plus souvent activable à la main avec le paramètre **--enable-xxx** où *xxx* représente la fonctionnalité en question. Lisez le changelog de PHP (Lien32) pour plus d'informations.

### 4.1. Avant de commencer

Halte ! si vous lancez votre commande maintenant, elle

échouera. Ceci pour la simple et bonne raison que notre distribution Ubuntu Server ne dispose pas des logiciels et bibliothèques C suffisants pour compiler PHP, le tout par défaut.
Nous devons installer un environnement de compilation, ainsi que la *libxml2*, car le support de XML est par défaut inclus dans PHP (--disable-xml est présente si on veut supprimer les possibilités XML de PHP et donc sa dépendance envers la libxml2).
Concernant tout le reste : Ubuntu Server possède tout ce qu'il faut par défaut.

Installation des outils et dépendances de base
```
~/monphp/src/php-5.3.2> sudo aptitude install
build-essential libxml2-dev autoconf2.13
```

**build-essential** est un métapaquet sous Ubuntu qui comporte tout l'environnement GNU de compilation (notamment le très connu GCC, mais pas seulement).
La *libxml2* se trouve dans le paquet **libxml2-dev** et nous rappelons qu'elle est nécessaire car par défaut (sans préciser au *configure*) le support de XML est activé dans PHP et tire ses fonctionnalités de bas niveau de la libxml2.
**autoconf2.13** est un paquet Linux aidant à la compilation, il possède des dépendances qui seront téléchargées par le gestionnaire de paquets, notamment *automake*, *libtool* et tout le tralala classique concernant la compilation de code C sous Linux. Ce sont des bibliothèques additionnelles très largement utilisées sous Linux permettant de charger des bibliothèques partagées, de créer des *scripts configure* automatiquement, etc. Attention, la version 2.13 est obligatoire et pas une autre (il en existe une plus récente)
Il est très fortement recommandé que vous vous familiarisiez un peu plus avec ces outils là si vous voulez comprendre en profondeur la compilation de PHP. Son manuel indique d'ailleurs ces dépendances (Lien33).

Pourquoi "*-dev*" alors que le paquet *libxml2* existe aussi et semble plus logique ? Tout simplement parce que les paquets suffixés par *-dev* contiennent les fichiers en-têtes C (fichiers .h) dont le compilateur aura besoin pour compiler les fonctions que le code utilise. Le paquet "normal" (sans le -dev) contient lui la bibliothèque partagée *.so* qui sera nécessaire à PHP pour démarrer. Télécharger le "-dev" récupèrera obligatoirement par dépendance le paquet "normal" contenant le module .so.
Ainsi, lorsque le *script configure* échoue car il manque une dépendance, très souvent on note le nom de cette dépendance, on cherche le paquet avec *aptitude search* et on trouve un paquet avec le bon nom suffixé par "-dev", puis on l'installe. Sous Ubuntu : c'est aussi simple que cela (avec quelques exceptions), car le gestionnaire aptitude gère très bien les dépendances et rapatrie tout ce qu'il faut pour nous.

### 4.2. Configuration, compilation, installation

Une fois l'installation de ces paquets terminée, nous pouvons lancer notre commande :

Lancement de la configuration des sources
```
~/monphp/src/php-5.3.2> ./configure
--prefix=/home/julien/monphp
```

Le *script configure* vérifie alors que votre système est compatible et possède tout ce qu'il faut pour compiler correctement par la suite, avec les options indiquées, c'est-à-dire rien du tout (compilation basique) sauf le chemin vers lequel installer PHP une fois celui-ci compilé.
Si tout se passe bien (et ça devrait être le cas), le script termine avec un message ressemblant à ceci :

fin du configure avec succès
```
+--------------------------------------------+
| License:                                   |
|                                            |
| This software is subject to the PHP License,
available in this      |
| distribution in the file LICENSE.  By
continuing this installation |
| process, you are bound by the terms of this
license agreement.     |
| If you do not agree with the terms of this
license, you must abort |
| the installation process at this point.
|
+--------------------------------------------+

Thank you for using PHP.

~/monphp/src/php-5.3.2>
```

Nous pouvons lancer la compilation à proprement parler avec la commande *make* qui va "fabriquer" notre projet et, entre autres, compiler les sources préparées par *configure*. L'étape prend plus ou moins de temps en fonction des capacités matérielles de votre machine mais aussi des options précisées : en installation par défaut, ce n'est pas très long, mais rien que le support XML prend environ 15-20 % du temps de la compilation (tout de même).

Compilation
```
~/monphp/src/php-5.3.2> make
```

Là, tout va bien se passer : en général si *configure* termine bien, le *make* terminera bien, sauf cas rares (erreurs dans le code source C, dans le makefile ou autre...).

Compilation terminée avec succès !
```
Build complete.
Don't forget to run 'make test'.
~/monphp/src/php-5.3.2>
```

Si vous voulez tester PHP sur votre plateforme avec vos options de configuration et envoyer un rapport de tests au PHPGroup (et ainsi contribuer modestement à l'amélioration de PHP), lancez la commande *make test*. Sinon, passons à l'installation avec *make install*.

Installation
```
~/monphp/src/php-5.3.2> make install
{ ... ... ... }

Installing PHP SAPI module:       cgi
Installing PHP CGI binary:
/home/julien/monphp/bin/
Installing PHP CLI binary:
/home/julien/monphp/bin/
Installing PHP CLI man page:
/home/julien/monphp/man/man1/
Installing build environment:
/home/julien/monphp/lib/php/build/
Installing header files:
/home/julien/monphp/include/php/
```

```
Installing helper programs:
/home/julien/monphp/bin/
  program: phpize
  program: php-config
Installing man pages:
/home/julien/monphp/man/man1/
  page: phpize.1
  page: php-config.1
Installing PEAR environment:
/home/julien/monphp/lib/php/
[PEAR] Archive_Tar    - already installed: 1.3.3
[PEAR] Console_Getopt - already installed: 1.2.3
[PEAR] Structures_Graph- already installed: 1.0.2
[PEAR] XML_Util       - already installed: 1.2.1
[PEAR] PEAR           - already installed: 1.9.0
Wrote PEAR system config file at:
/home/julien/monphp/etc/pear.conf
You may want to add: /home/julien/monphp/lib/php
to your php.ini include_path
/home/julien/monphp/src/php-5.3.2/build/shtool
install -c ext/phar/phar.phar
/home/julien/monphp/bin
ln -s -f /home/julien/monphp/bin/phar.phar
/home/julien/monphp/bin/phar
Installing PDO headers:
/home/julien/monphp/include/php/ext/pdo/
```

*make* nous informe des endroits dans lesquels il a installé PHP et ses utilitaires.

### 4.3. Vérification de l'arborescence installée

Voyons voir la structure installée de plus près ; installez l'utilitaire *tree* et lancez-le depuis le répertoire de base :

```
~/monphp/src/php-5.3.2> tree -L 3
.
|-- bin
|   |-- pear
|   |-- peardev
|   |-- pecl
|   |-- phar -> /home/julien/monphp/bin/phar.phar
|   |-- phar.phar
|   |-- php
|   |-- php-cgi
|   |-- php-config
|   `-- phpize
|-- etc
|   `-- pear.conf
|-- include
|   `-- php
|       |-- TSRM
|       |-- Zend
|       |-- ext
|       |-- include
|       `-- main
|-- lib
|   `-- php
|       |-- Archive
|       |-- Console
|       |-- OS
|       |-- PEAR
|       |-- PEAR.php
|       |-- PEAR5.php
|       |-- Structures
|       |-- System.php
|       |-- XML
|       |-- build
|       |-- data
|       |-- doc
|       |-- pearcmd.php
|       |-- peclcmd.php
|       `-- test
|-- man
|   `-- man1
|       |-- php-config.1
|       |-- php.1
|       `-- phpize.1
`-- src
    |-- php-5.3.2
    |   |-- CODING_STANDARDS
    |   [ ... toute la source ... ]
```

**/bin** contient les binaires et scripts exécutables : php (le SAPI CLI), php-cgi (le SAPI CGI), phar (un script de gestion des archives Phar), Pear et Pecl que nous connaissons et enfin deux scripts importants pour la suite, phpize et php-config (nous y reviendrons).
**/etc** contient la configuration (de pear pour le moment).
**/lib** contient les bibliothèques : ici il s'agit de l'installation minimale de PEAR, mais vous pouvez aussi y loger votre framework préféré (par exemple). Aussi, ce dossier contient les extensions PHP que vous voudrez par la suite ajouter. Il s'agit de */home/julien/monphp/lib/php/extensions/no-debug-non-zts-20090626* par défaut mais ça se change dans le php.ini.
**/man** contient les pages man, mais il faut les ajouter ou les lier au manuel de base, car celui-ci ne cherchera pas dans ce dossier-là par défaut.
**/include** contient les fichiers d'en-têtes C (fichiers .h) utilisés pour compiler PHP et qui pourront servir à tout programme futur voulant intégrer PHP ou une de ses fonctionnalités (les extensions PHP par exemple).

Vous pouvez personnaliser chacun de ces dossiers grâce aux options du *configure*, regardez la section *Directory and file names*.

Par défaut, aucun php.ini n'est généré. PHP va chercher par défaut dans *monphp/lib* ce qui n'est pas tout à fait correct, *monphp/etc* étant plus logique : nous changerons cela dans un chapitre relatant de la compilation avancée.
Sinon, les sources de PHP sont livrées avec deux fichiers php.ini, un pour le développement et un pour la production. Ils comportent tous les deux des options recommandées par le PHPGroup dans chacun des cas respectifs. Vous pouvez donc copier l'un d'entre eux (php.ini-development recommandé pour essayer) dans *monphp/lib* en le nommant *php.ini* et PHP l'utilisera immédiatement.

Le dossier des futures extensions que PHP chargera est */home/julien/monphp/lib/php/extensions/no-debug-non-zts-20090626*. La dernière partie du dossier précise que PHP a été compilé sans le mode debug, sans la gestion des threads et avec une API de son moteur portant le numéro 20090626.
Pour le changer, utilisez la directive *extension_dir* de php.ini.

### 4.4. Rappel de quelques commandes et outils utiles

Avant de continuer vers une étape de personnalisation de la compilation de PHP, rappelons quelques commandes et outils intéressants.
**php --ini** liste des informations sur le fichier php.ini utilisé.

informations sur le php.ini

```
~/monphp/bin> ./php --ini
Configuration File (php.ini) Path:
/home/julien/monphp/lib
Loaded Configuration File:         (none)
Scan for additional .ini files in: (none)
Additional .ini files parsed:      (none)
```

Compiler PHP avec l'option *--with-config-file-scan-dir=mon/dossier* fera que PHP cherchera dans *mon/dossier* tous les fichiers .ini qu'il trouvera et tentera de les charger. C'est très pratique pour éclater sa configuration en plusieurs fichiers .ini. J'utilise souvent php.ini puis autant de fichiers .ini annexes que d'extensions que j'ai : xdebug.ini, pdo.ini...
Attention au coût sur les performances au chargement de PHP, plus il y a de fichiers, plus c'est long.

**ldd** est un programme Linux qui liste les bibliothèques dont dépend un binaire pour être lancé (entre autres choses), appliquons-le sur PHP :

liste des bibliothèques partagées dont PHP a besoin

```
~/monphp/bin> ldd ./php
linux-gate.so.1 =>  (0x005a3000)
libcrypt.so.1 => /lib/tls/i686/cmov/libcrypt.so.1
(0x00ae5000)
libresolv.so.2 =>
/lib/tls/i686/cmov/libresolv.so.2 (0x0039a000)
librt.so.1 => /lib/tls/i686/cmov/librt.so.1
(0x00f9b000)
libm.so.6 => /lib/tls/i686/cmov/libm.so.6
(0x002d5000)
libdl.so.2 => /lib/tls/i686/cmov/libdl.so.2
(0x00f21000)
libnsl.so.1 => /lib/tls/i686/cmov/libnsl.so.1
(0x006fa000)
libxml2.so.2 => /usr/lib/libxml2.so.2
(0x00d54000)
libc.so.6 => /lib/tls/i686/cmov/libc.so.6
(0x00110000)
libpthread.so.0 =>
/lib/tls/i686/cmov/libpthread.so.0 (0x0092d000)
/lib/ld-linux.so.2 (0x00bda000)
libz.so.1 => /lib/libz.so.1 (0x00cd1000)
```

Notez que *lixml2.so* est présent dans le paquet *libxml2* : PHP a bien besoin de cette bibliothèque pour fonctionner, sa suppression entraînera l'impossibilité de lancer PHP (il lui manquera du code : la bibliothèque est partagée).
Si vous découvrez *ldd*, alors je ne saurais que vous conseiller de vous documenter (Lien34) vivement sur l'architecture des programmes Linux ELF (Executable Linking Format) et sur la compilation de code C de manière générale sous Linux, notamment le fonctionnement des linkers (Lien35).

**php -i | grep extension_dir** : le dossier qu'utilise PHP pour charger ses extensions

Le dossier des extensions PHP

```
~/monphp/bin> ./php -i | grep extension_dir
extension_dir =
/home/julien/monphp/lib/php/extensions/no-debug-
non-zts-20090626
```

**php -m** liste toutes les extensions chargées (statiquement - c'est-à-dire incluses à la compilation dans PHP - ou dynamiquement, en cherchant dans le *extension_dir*)

Le dossier des extensions PHP

```
~/monphp/bin> ./php -m
[PHP Modules]
Core
ctype
date
dom
ereg
fileinfo
filter
hash
iconv
json
libxml
pcre
PDO
pdo_sqlite
Phar
posix
Reflection
session
SimpleXML
SPL
SQLite
sqlite3
standard
tokenizer
xml
xmlreader
xmlwriter

[Zend Modules]
```

*PHP Modules* sont les extensions pour PHP. "Core" et "standard" représentent le cœur dont nous parlions dans les chapitres précédents : les fonctions string, array, la gestion des objets...
*Zend Modules* sont les débogueurs (ou les extensions qui redéfinissent des fonctions du Zend Engine, donc affectant très fortement les entrailles de PHP).
Notez que nous retrouvons là les extensions que nous avons compilées : nous n'avons rien précisé à *configure*, nous sommes donc en présence des extensions que le PHPGroup considère comme "par défaut pour cette version de PHP".

**php -v** renseigne sur la version de PHP, la SAPI utilisée, la date de compilation, la version du moteur et ses options (thread safe, debug activé, memory manager activé, etc.).

Version de PHP et SAPI

```
~/monphp/bin> ./php -v
PHP 5.3.2 (cli) (built: Mar 11 2010 20:09:34)
Copyright (c) 1997-2010 The PHP Group
Zend Engine v2.3.0, Copyright (c) 1998-2010 Zend
Technologies
```

**php --re foo** donne les fonctionnalités ajoutées à PHP par l'extension foo (si celle-ci est chargée, sinon erreur).

Liste des fonctionnalités de l'extension json

```
~/monphp/bin> ./php --re json
Extension [ <persistent> extension #12 json
version 1.2.1 ] {

  - Constants [10] {
```

```
    Constant [ integer JSON_HEX_TAG ] { 1 }
    Constant [ integer JSON_HEX_AMP ] { 2 }
    Constant [ integer JSON_HEX_APOS ] { 4 }
    Constant [ integer JSON_HEX_QUOT ] { 8 }
    Constant [ integer JSON_FORCE_OBJECT ] { 16 }
    Constant [ integer JSON_ERROR_NONE ] { 0 }
    Constant [ integer JSON_ERROR_DEPTH ] { 1 }
    Constant [ integer
JSON_ERROR_STATE_MISMATCH ] { 2 }
    Constant [ integer JSON_ERROR_CTRL_CHAR ] { 3
}
    Constant [ integer JSON_ERROR_SYNTAX ] { 4 }
  }

  - Functions {
    Function [ <internal:json> function
json_encode ] {

      - Parameters [2] {
        Parameter #0 [ <required> $value ]
        Parameter #1 [ <optional> $options ]
      }
    }
    Function [ <internal:json> function
json_decode ] {

      - Parameters [3] {
        Parameter #0 [ <required> $json ]
        Parameter #1 [ <optional> $assoc ]
        Parameter #2 [ <optional> $depth ]
      }
    }
    Function [ <internal:json> function
json_last_error ] {

      - Parameters [0] {
      }
    }
  }
}
```

Attention si vous voulez regarder les extensions *Core* ou *standard*, utilisez un pipe vers un programme tel que *less* parce que la sortie sera très grande.

**php --ri foo** liste les directives de configuration et des informations générales sur l'extension foo.

```
~/monphp/bin> ./php --ri json

json

json support => enabled
json version => 1.2.1
```

Ces quelques outils et commandes rappelés, nous pouvons nous lancer maintenant dans une compilation et une personnalisation plus avancées.

Vous noterez au passage l'extrême modularité du langage PHP dont on connaît la réputation. Le design de PHP et de son moteur Zend Engine sont très flexibles, et c'est en se penchant sur la compilation du programme qu'on le remarque particulièrement d'un point de vue extérieur (plongez dans son code source sinon, si vous en avez les capacités).

## 5. Installation avancée

Dans ce chapitre, nous verrons quelques unes des nombreuses directives du *script configure*, leur utilisation et leur impact sur le binaire PHP compilé par la suite.

A chaque invocation du *script configure* (avec ses paramètres), celui-ci enregistre la ligne de commande tapée dans un fichier appelé 'config.nice', ce qui permet de relancer la commande ou de la modifier en éditant ce fichier

### 5.1. PHP minimal

L'option **--disable-all** permet de compiler un PHP minimal, avec uniquement les modules essentiels. C'est un raccourci pour éviter d'écrire tous les *–disable-xxx*
Les modules chargés sont :

```
~/monphp/bin> ./php -m
Core
date
ereg
pcre
Reflection
SPL
standard
```

Ceci est valable pour PHP5.3, *Reflection* et *SPL* sont devenus obligatoires en 5.3, on pouvait les désactiver dans la branche 5.2
Votre PHP possède une empreinte mémoire minimale (les possibilités avancées pour réduire encore cette empreinte sont en dehors du cadre de cet article), un temps de démarrage amélioré, mais en contrepartie il ne saura pas faire grand chose

Quelques statistiques sur un PHP minimal

```
~/monphp/bin> ./php -r 'echo
count(current(get_defined_functions()));'
647

~/monphp/bin> ./php -r 'echo
count(get_declared_classes());'
69

~/monphp/bin> ./php -r 'echo
count(get_defined_constants());'
417
```

PHP minimal ne comporte aucune fonctionnalité d'accès à des bases de données, aucune gestion de XML, pas de support des sessions... Des programmes comme PEAR ne fonctionneront pas.

On utilise un PHP minimal lorsqu'on développe des extensions que l'on souhaite tester (pour limiter les risques d'interaction), ou lorsqu'on crée des démons qui n'ont besoin que des fonctions de base, mais d'une empreinte mémoire maîtrisée. Le temps de démarrage de PHP est très optimisé aussi, cela peut servir pour des scripts à haute charge ayant besoin de forker.

### 5.2. Plusieurs PHP

Si vous voulez compiler plusieurs PHP, vous devrez alors changer leurs préfixes, c'est le plus pratique à gérer.

Il existe une autre solution : **--program-prefix=PREFIX** donnera naissance à des binaires comme *PREFIXphp* et inversement, **--program-suffix=SUFFIX** fera que make créera des binaires comme *phpSUFFIXE* .

utilisation d'un suffixe de programme '532', pour rappeler la version 5.3.2

```
~/monphp/src/php-5.3.2> ./configure
--prefix=/home/julien/monphp --program-suffix=532

[Après compilation, tous les binaires ont un nom
suffixé par "532"]
~/monphp/bin> ./pear532
```

En réalité, si vous compilez les extensions statiquement (nous allons y venir), seul le binaire de PHP (généralement le CLI) est utile : c'est lui qui possède toutes les fonctionnalités compilées.
Si vous voulez compiler plusieurs versions simplement, vous pouvez vous pencher sur le projet phpfarm (Lien36) : c'est un ensemble de scripts qui fournissent un frontend un peu plus sympa.

### 5.3. Compiler des extensions (fournies)

Si on passe par la compilation manuelle (l'objet de cet article), c'est en grande partie pour compiler les extensions que l'on souhaite et obtenir un PHP "sur mesure". Nous allons donc voir comment intégrer des extensions dans PHP, à la compilation.
Tout se passe une fois encore avec le *script configure*.
**--enable-xxx** active l'extension xxx pour une compilation statique (extension fusionnée).
**--enable-xxx=shared** active l'extension xxx pour une compilation dynamique (objet .so construit et installé dans le dossier des extensions).
**--with-xxx** active l'extension xxx pour une compilation statique (extension fusionnée). Attention, cette extension possède une dépendance envers une/plusieurs bibliothèques tierces.
**--with-xxx=shared** active l'extension xxx pour une compilation dynamique (objet .so construit et installé dans le dossier des extensions). Attention, cette extension possède une dépendance envers une/plusieurs bibliothèques tierces.
**--disable-xxx** désactive l'extension xxx qui serait sinon incluse de manière statique.

Lisez le détail de *./configure --help* pour plus d'informations sur certaines options. Tout y est écrit clairement.

Activer une extension avec l'option "with" requiert une ou plusieurs bibliothèques qui ne sont pas forcément présentes sur le système. Gardez en tête une technique simple : notez le nom de la bibliothèque dans le message d'erreur de *configure*, cherchez-la dans le gestionnaire de paquets *aptitude search xxx* et notez le nom du paquet qui convient, si xxx-dev est disponible, il faut prendre celui-là.

Une installation bien personnalisée

```
~/monphp/src/php-5.3.2> ./configure
--prefix=/home/julien/monphp --disable-cgi
--with-config-file-scan-
dir=/home/julien/monphp/etc --disable-short-tags
--with-zlib --with-bz2
--with-curl --enable-exif --with-gd --enable-gd-
native-ttf --enable-intl --enable-mbstring
--with-mcrypt --with-mysqli=mysqlnd
--with-pdo-mysql=mysqlnd --with-libedit --with-
openssl --enable-soap=shared --enable-sqlite-utf8
--with-tidy=shared --enable-wddx --with-xsl
--enable-zip
```

Le premier message d'erreur apparaît :

Il manque une bibliothèque sur le système

```
[...]
checking for BZip2 in default path... not found
configure: error: Please reinstall the BZip2
distribution
~/monphp/src/php-5.3.2>
```

Qui se solutionne par :

```
~/monphp/src/php-5.3.2> sudo aptitude install
libbz2-dev
```

Installez la bibliothèque dans sa version développement. Sous Ubuntu Server, le paquet de la bibliothèque est déjà installé le plus souvent (sinon il sera inclus par dépendance), mais pas le paquet de développement (*xxx-dev*) nécessaire pour que le compilateur puisse lier les fonctions.

Avec la ligne de *configure* de l'exemple ci-dessus, PHP est hautement personnalisé et beaucoup de fonctionnalités lui sont ajoutées, beaucoup d'entre elles de manière statique car elles seront utilisées souvent (dans l'exemple), quelques-unes sous forme dynamique (shared object).
Notez que vous nécessiterez l'ajout d'un nombre assez conséquent de bibliothèques de code si vous suivez cette ligne.
Notez aussi que les extensions dynamiques sont à activer dans un php.ini avec la directive *extension=xxx*, le dossier dans lequel PHP cherche les fichiers ini a été précisé dans l'exemple grâce à l'option *--with-config-file-scan-dir=/home/julien/monphp/etc*.

Pour relancer la ligne de *configure*, jouez avec l'historique de votre interpréteur (bash) ou lancez le script *config.nice*.

```
~/monphp/src/php-5.3.2> sudo aptitude install
libcurl4-dev libpng-dev libjpeg-dev libicu-dev
libmcrypt-dev libedit-dev libtidy-dev libxslt-dev
libssl-dev
~/monphp/src/php-5.3.2> ./config.nice
```

Nous choisissons la solution simple : le gestionnaire de paquets. Si vous préférez télécharger la bibliothèque sur le site officiel de celle-ci, libre à vous.
Vous devrez alors préciser son chemin grâce aux nombreuses options que propose le *script configure* à cet effet, lisez son aide. Il peut être intéressant de télécharger la bibliothèque soi-même dans le cas où votre gestionnaire de paquets ne la prend pas en charge, ou n'a pas la bonne version. Ce n'est jamais (sauf si vous tirez sur les ficelles) le cas sous Ubuntu Server. De plus, il est relativement casse-tête d'installer soi-même des bibliothèques manuellement, du moins si vous ne maitrisez pas bien Linux et votre distribution ; n'oubliez pas que vous devrez aussi compiler et installer ladite bibliothèque à la main ainsi que gérer ses mises à jour et éventuellement sa liaison avec le système...

Toutes les extensions ne peuvent être compilées et chargées de manière dynamique, certaines nécessitent une liaison statique.

Lorsque le *script configure* se termine correctement, vous enchaînez bien sûr par une compilation via **make**, puis l'installation à proprement parler par **make install**.

Attention, veillez à bien nettoyer votre espace avant chaque compilation (make) par la commande **make clean**. Si vous oubliez cette étape, il est possible que la compilation échoue, ou pire fonctionne, mais pas le PHP compilé. *make* mémorise des données afin de ne pas recompiler chaque module entre deux compilations, mais ceci est souvent cause de problèmes casse-tête. Renseignez-vous sur l'outil *make* pour plus de détails.

### 5.3.1. Vérifier la présence des extensions

Maintenant que notre PHP est fabriqué et installé, nous pouvons relancer un **ldd** :

Vérification des dépendances du binaire php

```
~/monphp/bin> ldd ./php
linux-gate.so.1 =>  (0x00ccd000)
libcrypt.so.1 => /lib/tls/i686/cmov/libcrypt.so.1 (0x00110000)
libz.so.1 => /lib/libz.so.1 (0x007fc000)
libexslt.so.0 => /usr/lib/libexslt.so.0 (0x00a2f000)
libedit.so.2 => /usr/lib/libedit.so.2 (0x001d1000)
libncurses.so.5 => /lib/libncurses.so.5 (0x00b6e000)
librt.so.1 => /lib/tls/i686/cmov/librt.so.1 (0x00b4c000)
libmcrypt.so.4 => /usr/lib/libmcrypt.so.4 (0x00323000)
libltdl.so.7 => /usr/lib/libltdl.so.7 (0x00142000)
libpng12.so.0 => /usr/lib/libpng12.so.0 (0x008c9000)
libbz2.so.1.0 => /lib/libbz2.so.1.0 (0x006b8000)
libm.so.6 => /lib/tls/i686/cmov/libm.so.6 (0x00d2c000)
libdl.so.2 => /lib/tls/i686/cmov/libdl.so.2 (0x00459000)
libnsl.so.1 => /lib/tls/i686/cmov/libnsl.so.1 (0x00c87000)
libssl.so.0.9.8 => /lib/i686/cmov/libssl.so.0.9.8 (0x0014c000)
libcrypto.so.0.9.8 => /lib/i686/cmov/libcrypto.so.0.9.8 (0x0045d000)
libcurl-gnutls.so.4 => /usr/lib/libcurl-gnutls.so.4 (0x00192000)
libicui18n.so.40 => /usr/lib/libicui18n.so.40 (0x00d52000)
libicuuc.so.40 => /usr/lib/libicuuc.so.40 (0x006ca000)
libicudata.so.40 => /usr/lib/libicudata.so.40 (0xb6a88000)
libicuio.so.40 => /usr/lib/libicuio.so.40 (0x001f4000)
libxslt.so.1 => /usr/lib/libxslt.so.1 (0x001ff000)
libxml2.so.2 => /usr/lib/libxml2.so.2 (0x008f1000)
libgcc_s.so.1 => /lib/libgcc_s.so.1 (0x00ad3000)
libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0x00f94000)
libstdc++.so.6 => /usr/lib/libstdc++.so.6 (0x00355000)
libresolv.so.2 => /lib/tls/i686/cmov/libresolv.so.2 (0x00236000)
libgcrypt.so.11 => /lib/libgcrypt.so.11 (0x00837000)
libbsd.so.0 => /lib/libbsd.so.0 (0x0024a000)
libpthread.so.0 => /lib/tls/i686/cmov/libpthread.so.0 (0x00c52000)
/lib/ld-linux.so.2 (0x00f77000)
libidn.so.11 => /usr/lib/libidn.so.11 (0x00253000)
liblber-2.4.so.2 => /usr/lib/liblber-2.4.so.2 (0x00286000)
libldap_r-2.4.so.2 => /usr/lib/libldap_r-2.4.so.2 (0x00295000)
libgssapi_krb5.so.2 => /usr/lib/libgssapi_krb5.so.2 (0x002e0000)
libgnutls.so.26 => /usr/lib/libgnutls.so.26 (0x005a3000)
libgpg-error.so.0 => /lib/libgpg-error.so.0 (0x0030a000)
libsasl2.so.2 => /usr/lib/libsasl2.so.2 (0x0064b000)
libkrb5.so.3 => /usr/lib/libkrb5.so.3 (0x00ba6000)
libk5crypto.so.3 => /usr/lib/libk5crypto.so.3 (0x00665000)
libcom_err.so.2 => /lib/libcom_err.so.2 (0x00f2a000)
libkrb5support.so.0 => /usr/lib/libkrb5support.so.0 (0x0030f000)
libkeyutils.so.1 => /lib/libkeyutils.so.1 (0x00317000)
libtasn1.so.3 => /usr/lib/libtasn1.so.3 (0x00447000)
```

Naturellement, PHP est maintenant lié au système via beaucoup plus de bibliothèques partagées que lors de son installation minimale. Comme nous avons compilé la majorité de ses extensions statiquement, il n'est plus possible de les enlever et il faudra veiller à ce que ces dépendances soient systématiquement satisfaites.

La compilation avec liaison statique des dépendances envers les bibliothèques du système dépasse le cadre de cet article. La plupart du temps, c'est un casse-tête monstrueux qui se solde par un échec.
PHP dépend (dans notre cas) de beaucoup de bibliothèques externes, le système de nommage et le loader (ld) devraient en théorie, en assurer la présence et la mise à jour.

Voyons le contenu du dossier des extensions de PHP :

Vérification des extensions dynamiques de PHP

```
~/monphp/lib/php/extensions/no-debug-non-zts-20090626> ls -l
-rwxr-xr-x 1 julien julien 1041408 2010-03-14 16:18 soap.a
-rwxr-xr-x 1 julien julien  840564 2010-03-14 16:18 soap.so
-rwxr-xr-x 1 julien julien  125350 2010-03-14 16:18 tidy.a
-rwxr-xr-x 1 julien julien  116434 2010-03-14 16:18 tidy.so
```

Bien, toutes les extensions que nous avions indiquées via le suffixe *"=shared"* sont bien partagées et seront liées dynamiquement au lancement de PHP si php.ini les précise.

Par la même occasion, nous pourrions regarder leurs dépendances :

Les dépendances systèmes des extensions PHP
```
~/monphp/lib/php/extensions/no-debug-non-zts-20090626> ldd tidy.so
linux-gate.so.1 =>  (0x0067d000)
libtidy-0.99.so.0 => /usr/lib/libtidy-0.99.so.0 (0x00735000)
libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0x00384000)
/lib/ld-linux.so.2 (0x0083b000)
```

Charger tidy.so dans PHP nécessitera bien la *libtidy* sur le système, nous l'avions installée avant de compiler.
Le fichier .a est une archive contenant les fichiers .o qui représentent du code liable (relocatable) mais non chargeable. Ces fichiers ne vous serviront pas pour utiliser PHP, vous pouvez tranquillement les effacer. Pour plus d'informations à leur sujet, renseignez-vous sur le format des exécutables Linux ELF.

### 5.4. Quelques options de compilation

Là encore votre connaissance de Linux, des bibliothèques open source C que l'on peut trouver et de leur utilité vous feront gagner un temps précieux.
Par exemple **--with-libedit** ou **--with-readline** permettent de bénéficier d'un mode interactif (php -a) beaucoup plus complet : historique des commandes, complétions des fonctions PHP... Bash ou encore ssh utilisent ces bibliothèques, par exemple :

PHP interactif aidé de la libedit
```
~/monphp/bin> ./php -a
Interactive shell

php > str_  [pression sur tab]
str_getcsv      str_ireplace    str_pad
str_repeat      str_replace     str_rot13
str_shuffle     str_split       str_word_count
```

**--enable-debug** active le mode de débogage. Principalement, cela active le switch **-g** de GCC qui va alors laisser les symboles de débogage dans le binaire ELF. Il sera donc utilisable au travers d'un débogueur comme *gdb*. Cela est indispensable dans le cas du développement de PHP lui-même ou d'une de ses extensions.
Le résultat de **php -v** indiquera *(DEBUG)* et certaines extensions devront être recompilées avec ce mode pour être utilisables (les débogueurs PHP notamment comme Xdebug).
Le mode debug désactive aussi toutes les optimisations du compilateur avec un switch **-O0** automatiquement.
Pour déboguer la mémoire, dmalloc peut être utilisé avec l'option **--enable-dmalloc**

**--enable-embed** permet de construire la SAPI embed, qui permet de lier PHP dans son code C grâce à libphp5.so qui sera construit lors de la compilation.

Exemple minimal simplifié d'inclusion de PHP dans du code C
```
#include <php_embed.h>

int main(int argc, char *argv[])
{
    PHP_EMBED_START_BLOCK(argc, argv)
        zend_eval_string("echo 'Hello World!';",
NULL,
                         "My Hello World"
TSRMLS_CC);
    PHP_EMBED_END_BLOCK()
    return 0;
}
```

**--with-curlwrappers** : lie les couches basses du réseau de PHP sur curl plutôt que leur implémentation native.

Vérification des couches basses du réseau pour PHP
```
~/monphp/bin> ./php -r
'var_dump(stream_get_meta_data(fopen("http://www.
google.fr", "r")));'
array(10) {
  ## tronqué ##
  ["wrapper_type"]=>
  string(4) "cURL"
  ["stream_type"]=>
  string(4) "cURL"
  ## tronqué ##
```

**--with-mysqli=mysqlnd | --with-pdo-mysql=mysqlnd** : depuis PHP5.3, les fonctionnalités MySQL de PHP peuvent utiliser MySQLnd (NativeDriver). NativeDriver est une couche basse de dialogue avec MySQL développée par le PHPGroup, et permettant notamment une licence d'utilisation identique à celle de PHP, ainsi que des optimisations sur les structures mémoires sous-jacentes. Si vous ne précisez pas mysqlnd, l'extension sera liée avec la libmysql, vous devrez donc télécharger son paquet. MySQLnd est pratique en ce sens que PHP n'est plus du tout dépendant de MySQL pour être compilé.

**--with-zend-vm=switch|goto|call** : pour les puristes, elle change le modèle d'exécution de la machine virtuelle PHP. Cela affecte le moteur d'exécution de la machine virtuelle. Les plus téméraires pourront se renseigner dans la source directement (Lien37) ou lire quelques benchmarks ici (Lien38).

## 6. Installation personnalisée d'extensions

Bien, nous allons voir ici comment installer des extensions dynamiques pour PHP une fois celui-ci compilé et installé. Nous allons aussi voir comment lier celles-ci de manière statique à PHP, il faudra pour cela entièrement le recompiler.

### 6.1. Dynamiquement

Imaginons que nous souhaitions installer l'extension **Memcache**, qui n'est pas livrée avec PHP. Nous procéderons en trois temps.

1. Télécharger et dépaqueter la source de l'extension (dans le dossier monphp/src afin de ranger les choses).
2. Importer l'environnement de préparation des sources qui permettra de lier le code de l'extension au code source de PHP.
3. Configurer, compiler, installer et activer le .so obtenu dans php.ini.

Toutes ces étapes peuvent être exécutées en une seule

commande : **pecl install xxxx**. Nous n'utiliserons pas ce raccourci afin de détailler et pour que vous compreniez bien le cheminement.

Les paquets *autoconf-2.13*, *automake* et *libtool* sont indispensables pour préparer les sources d'une extension. L'installation d'autoconf-2.13 résout les autres dépendances.

Téléchargement et dépaquetage de l'extension

```
~/monphp/src> ../bin/pecl download memcache
downloading memcache-2.2.5.tgz ...
Starting to download memcache-2.2.5.tgz (35,981 bytes)
.........done: 35,981 bytes
File /home/julien/monphp/src/memcache-2.2.5.tgz downloaded

~/monphp/src> tar xzf memcache-2.2.5.tgz && cd memcache-2.2.5
~/monphp/src/memcache-2.2.5>
```

Nous nous trouvons dans le dossier des sources de l'extension Memcache. A ce stade-là, il est nécessaire d'importer les outils de configuration et de compilation dans l'arbre des sources.
C'est là qu'intervient le programme **phpize**. Il suffit de lancer *phpize* dans le dossier courant (celui des sources de l'extension) et il crée automatiquement *configure* (et autres outils nécessaires comme libtool, config.m4, etc.).

importation des outils de préparation et de compilation dans les sources

```
~/monphp/src/memcache-2.2.5> ../../bin/phpize
Configuring for:
PHP Api Version:         20090626
Zend Module Api No:      20090626
Zend Extension Api No:   220090626
~/monphp/src/memcache-2.2.5>
```

On voit bien que *phpize* a trouvé les versions des API internes à PHP : en effet il est possible que vous ayez à recompiler vos extensions lorsque vous changez de version de PHP (surtout sur les majeures) car les API internes changent.
Dès lors, vous pouvez lancer *configure* pour votre extension.

options de configuration d'une extension PHP

```
~/monphp/src/memcache-2.2.5> ./configure --help
[...]
--with-libdir=NAME      Look for libraries
in .../NAME rather than .../lib
--with-php-config=PATH  Path to php-config [php-config]
--enable-memcache       Enable memcache support
--disable-memcache-session       Disable memcache
session handler support
--with-zlib-dir[=DIR]   memcache: Set the path to
ZLIB install prefix.
--enable-debug          compile with debugging
symbols
--enable-shared[=PKGS]  build shared libraries
[default=yes]
--enable-static[=PKGS]  build static libraries
[default=yes]
--enable-fast-install[=PKGS]  optimize for fast
installation [default=yes]
--with-gnu-ld           assume the C compiler
uses GNU ld [default=no]
--disable-libtool-lock  avoid locking (might
break parallel builds)
--with-pic              try to use only PIC/non-
PIC objects [default=use both]
--with-tags[=TAGS]      include additional
configurations [automatic]
--with-gnu-ld           assume the C compiler
uses GNU ld [default=no]
```

L'aperçu ci-dessus a été tronqué, ce qui nous intéresse c'est ce qui est spécifique à l'extension. On devinera ici : **--enable-memcache** ou encore **–disable-memcache-session**.
Soyons clairs, il n'est pas nécessaire d'activer **--enable-memcache** lorsqu'on configure memcache... C'est idiot oui, mais ça provient de la *libtool* (tout comme les autres options que vous voyez et avez déjà vues). Evidemment, si on lance *configure* sans aucune option, il configurera les sources pour activer memcache, est-il besoin de le préciser ?

**--with-php-config** est important : il permet de préciser le chemin vers le script php-config. Ce script possède en lui toutes les variables qui reflètent la configuration de PHP qui a été effectuée (la manière dont il a été compilé). Préciser cela reviendra à faire en sorte que le *configure* trouve automatiquement le prefix (le chemin d'installation) notamment. Il est recommandé de le préciser (mais pas obligatoire).

Testez : lancez *monphp/bin/php-config* et regardez ce qui s'affiche : toute la configuration qui a été utilisée pour compiler PHP (les chemins notamment). Le *configure* des extensions peut utiliser ce script pour être aidé.

Comme vous le savez (j'espère), l'extension memcache peut aussi être compilée pour créer un backend pour les sessions PHP. Par défaut c'est le cas, si vous ne le souhaitez pas, activez **--disable-memcache-session**.

Nous configurons pour une compilation en module. La compilation générera donc un objet .so : une extension PHP, qu'il suffira de copier dans le dossier des extensions et d'activer dans php.ini (généralement, *make install* exécute ces étapes). Nous verrons la compilation statique dans le chapitre suivant.

configuration compilation, installation et activation de l'extension

```
~/monphp/src/memcache-2.2.5> ./configure --with-php-config=/home/julien/monphp/bin/php-config && make && make install
[...]

Build complete.
Don't forget to run 'make test'.
Installing shared extensions:
/home/julien/monphp/lib/php/extensions/debug-non-zts-20090626/
```

Et voilà : il faut ajouter manuellement la ligne *extension=memcache.so* dans le fichier ini utilisé si vous voulez utiliser l'extension.

Si vous voulez nettoyer votre source, exécutez **make clean**

puis **phpize --clean** : vous retrouverez les sources de votre extension telles qu'elles étaient juste après leur extraction.

### 6.2. Statiquement

Nous allons maintenant voir comment compiler une extension extérieure (non présente dans la source de PHP par défaut) de manière statique : c'est-à-dire fusionnée dans le binaire de PHP.
L'opération n'est pas complexe mais diffère un peu de la méthode dynamique, voici les étapes à suivre :

1. télécharger et dépaqueter la source de l'extension dans le dossier *source-de-php/ext*, là où sont logées les autres extensions, fournies par défaut ;
2. supprimer le *script configure* de la source PHP ;
3. lancer le *script buildconf* pour régénérer le *configure* et inclure le switch (--enable ou --with) concernant l'extension fraichement ajoutée ;
4. lancer le *configure* de PHP normalement, en ajoutant le switch concernant l'extension concernée.

Passons à la pratique, je propose de compiler statiquement l'extension APC (Lien39) dans PHP. C'est un cas pratique très classique : APC est un cache d'OPCode très proche de PHP et du ZendEngine. L'utiliser de manière dynamique (chargé comme un .so) n'est pas très malin, pour tirer partie de toute la puissance du cache d'opcode, il faut le fusionner à PHP.

Intégration d'une extension dans la source de PHP pour compilation statique

```
~/monphp/src> ../bin/pecl download apc-alpha
downloading APC-3.1.3p1.tar ...
~/monphp/src> tar xf APC-3.1.3p1.tar
~/monphp/src> mv APC-3.1.3p1 ./php-5.3.2/ext
```

Le dossier APC a été ajouté à la source de PHP dans son dossier ext/. Régénérons le *configure* :

Recréation du script configure pour prendre en compte APC

```
~/monphp/src> cd php-5.3.2
~/monphp/src/php-5.3.2> rm ./configure
~/monphp/src/php-5.3.2> ./buildconf --force
Forcing buildconf
rebuilding configure
rebuilding main/php_config.h.in
```

voilà ! regardons le *configure* :

Le configure prend bien APC en compte

```
~/monphp/src/php-5.3.2>./configure --help | grep
apc
--enable-apc            Enable APC support
--enable-apc-filehits   Enable per request file
info about files used from the APC cache (ie:
apc_cache_info('filehits'))
--disable-apc-mmap
--enable-apc-sem
--disable-apc-pthreadmutex
--enable-apc-spinlocks
```

Il ne reste plus qu'à recompiler PHP avec un *configure* qui inclut **--enable-apc** (je vous laisse vous pencher sur les autres options d'apc).

Oui, qui dit liaison statique dit recompiler tout PHP à chaque fois. C'est un des inconvénients de la compilation/liaison statique d'extension.

## 7. Petit tour d'extensions PHP sympathiques

Si vous n'avez jamais fouillé PECL, vous allez peut-être être étonnés par les extensions qui y sont présentes. Nous n'allons pas apprendre à utiliser PECL ici, c'est très simple, je vous laisse voir sa documentation (Lien40). Nous allons passer en revue quelques extensions intéressantes, et en plus vous savez les installer maintenant.

Rappel : une extension PHP, c'est du code C. Le code C compilé est environ 10 fois plus rapide que du code PHP (souvent plus, ~15 fois), il ne faut donc pas hésiter à utiliser une extension PHP pour des tâches que vous aviez l'habitude d'implémenter en PHP, surtout si le code est "lourd". Vous allez voir qu'on peut parser du bbcode, ou encore jouer avec du Yaml via des extensions PHP. Refaire cela en PHP va déjà vous prendre un temps non négligeable et sera beaucoup plus lent que l'appel des fonctions similaires apportées par les extensions en question, pour peu que celles-ci répondent précisément à votre besoin. Dans le cas contraire, vous pourrez songer à les modifier si vous vous sentez d'attaque.

Attention, ne déployez pas une extension non stable (alpha, bêta) en production, sauf si vous avez bien analysé son code source et que vous savez précisément ce que vous faites.

Voici un panaché choisi par mes soins, pour vous prouver que des extensions PHP... il y en a !
Regardez bien la date de la dernière version de l'extension et la version de PHP nécessaire pour la construire. Il peut arriver que certaines extensions ne soient plus maintenues depuis des années (mais fonctionnelles).

Certaines des extensions ne sont pas hébergées sur pecl.php.net, car leurs auteurs ont préféré les héberger eux-mêmes (ou utilisent une licence différente). A ce moment-là, il n'est pas possible d'utiliser l'utilitaire PECL, il faut télécharger l'extension manuellement, vous connaissez le reste de la procédure (voir les chapitres précédents).

Comme ce sont des "extensions", on utilise très souvent la notation "ext/xxx" pour désigner l'extension xxx dans une discussion ou un manuel. Par exemple, "utilisez ext/apc pour accélérer vos applications", ou encore "le support des sessions en PHP est implémenté par ext/session".

### 7.1. bytekit

Pour tous ceux qui sont intéressés par la représentation interne des instructions de la machine virtuelle de PHP ("l'OPcode"), ext/bytekit vous permet d'en prendre connaissance.
Cette extension propose aussi le tracé graphique du schéma d'exécution du code (branchements), ce qui permet de l'analyser en vue de le comprendre et éventuellement de fabriquer un optimiseur ou de créer des statistiques de code très poussées.
Une extension analogue est ext/vld (Virtual Logic Disassembler) (Lien41). Téléchargez ext/bytekit ici

(Lien42). Il peut être intéressant d'utiliser le frontend bytekit-cli avec (Lien43).

### 7.2. Imagick

Fournisseur de services vers la bibliothèque ImageMagick (Lien44). Cette bibliothèque permet une gestion avancée des images, une centaine de formats est gérée, dont DPX, EXR, GIF, JPEG, JPEG-2000, PDF, PhotoCD, PNG, Postscript, SVG, TIFF... Une bibliothèque indispensable si vous traitez massivement des images en PHP, elle est plus rapide et plus fournie que la bibliothèque GD (fournie avec PHP) et propose une orientation objets du code PHP l'utilisant. Sa documentation est sur php.net (Lien45).

### 7.3. Inclued

ext/inclued (notez l'orthographe) permet de tracer un graphe des appels à include/require. On peut donc voir graphiquement quel fichier inclut quel fichier, etc. (GraphViz nécessaire). Pour peu que vous manipuliez un projet très orienté objets - à base de framework notamment - et que l'autoload ne soit pas utilisé (les instructions require/include doivent être présentes), ext/inclued pourra servir pour tracer des graphes de dépendances objets à moindre frais. Très pratique.

### 7.4. Memcache

Nous l'avons utilisée comme exemple dans les chapitres précédents. ext/memcache fournit une API pour piloter un serveur Memcached. Vous trouverez plus d'informations sur le site officiel de Memcached (Lien46) et sur le manuel de ext/memcache (Lien47) présent dans php.net. Une extension très recommandée pour gérer la montée en charge d'applications par le cache d'entités (souvent des objets).

### 7.5. Scream

Lorsque la bibliothèque ext/scream est activée, l'effet de masquage des erreurs, obtenu par le caractère at (@) placé devant les commandes PHP, est annulé : les erreurs sont alors gérées normalement, comme en l'absence du caractère @.
Son manuel (très court, car ext/scream ne déploie aucune classe/fonction supplémentaire dans PHP) est sur php.net (Lien48).

### 7.6. bbcode

Je cite le manuel de PHP : "ext/bbcode se propose de vous aider dans l'analyse du texte BBCode afin de le convertir en HTML ou dans un autre langage à base de balises. Elle utilise une analyse en un seul passage et est bien plus rapide qu'une approche basée sur les expressions régulières. De plus, elle vous aidera à obtenir du code HTML valide en réorganisant les balises d'ouverture et de fermeture et en fermant automatiquement les balises."
Son manuel est disponible sur php.net (Lien49).

### 7.7. parsekit

ext/parsekit fait globalement la même chose que ext/bytekit, mais propose des fonctions PHP. Ainsi, elle propose deux fonctions prenant soit un fichier PHP, soit une chaîne représentant du code PHP et elle passe l'analyseur syntaxique PHP dessus. Un tableau à multiples dimensions est retourné contenant les OPCodes. Les erreurs (de tous types) sont gérées.
Le manuel de ext/parsekit fait partie du manuel de PHP sur php.net (Lien50).

### 7.8. yaml

ext/yaml utilise la libyaml (Lien51) pour permettre une conversion de PHP vers la syntaxe Yaml et inversement.
Son manuel est présent sur php.net (Lien52)

### 7.9. http

Si vous avez besoin d'un client HTTP puissant et rapide, ext/http est faite pour vous. Elle peut optionnellement utiliser Curl. ext/http est sans doute la manière la plus aboutie d'utiliser le protocole HTTP en PHP. La création d'un serveur ou d'un client est tout à fait dans ses cordes. Elle gère SSL, la négociation de contenu et de langues, les cookies, les jeux de caractères, le cache, la compression Gzip et le pooling de connexions. Elle peut aussi ajouter des fonctions de callback pour le buffer de sortie PHP. Rien que ça !
Son manuel est dans php.net (Lien53)

### 7.10. haru

ext/haru fournit des services vers la libharu (Lien54) : c'est une bibliothèque C libre permettant la génération de fichiers PDF. Nous sommes là en présence d'une extension écrite en C, donc dépassant largement les performances de n'importe quelle implémentation identique réalisée en PHP. Reste à tester les possibilités offertes, un article existe par ici pour les curieux (Lien55).
Son manuel se trouve sur php.net (Lien56)

### 7.11. amfext

ext/amfext permet d'implémenter le protocole d'Adobe AMF et AMF3. Voyez le site officiel pour plus d'informations (Lien57). Attention, ça n'a pas été testé sous PHP5.3 ni sous PHP5.2 pour Linux.

### 7.12. svn

ext/svn offre des possibilités d'interaction avec SVN côté PHP au travers de la libsvn. Toutes les opérations clientes sont rendues possibles au moyen de fonctions PHP. Indispensable si vous devez communiquer avec un serveur SVN depuis PHP.
Son manuel est sur php.net (Lien58).

### 7.13. xdiff

ext/xdiff implémente libxdiff pour PHP (Lien59). Vous pourrez créer des patchs (textes ou binaires), extraire des diffs ou exécuter des fusions (merge). Là encore, si vous utilisez PHP au sein de votre système d'informations, les possibilités xdiff sont à considérer.
Le manuel de ext/xdiff est intégré à php.net (Lien60).

### 7.14. sphinx

ext/sphinx ajoute via libsphinxclient les possibilités de Sphinx à PHP. Sphinx est un moteur de recherche SQL full text libre. Son site Web vous en apprendra plus si besoin (Lien61).
Le manuel de ext/sphinx est sur php.net (Lien62)

### 7.15. runkit

ext/runkit est un outil de sandboxing permettant la modification de fonctions, classes, constantes ou variables (y compris superglobales) au runtime PHP (c'est-à-dire directement pendant l'exécution). Une extension très puissante qui permet aussi de faire du vrai n'importe quoi si on est pas très averti (comme par exemple changer la définition d'une fonction, alors qu'elle est en cours d'utilisation). Voyez le manuel de ext/runkit intégré à php.net pour plus d'informations (Lien63).

### 7.16. xdebug

On ne présente plus l'extension que tout développeur devrait posséder. Xdebug permet de tracer la pile d'appel de fonctions PHP, d'afficher l'état de la mémoire et de récupérer des informations sur le moteur PHP via des fonctions.
Xdebug est aussi et avant tout un débogueur PHP très efficace, ainsi qu'un profileur. Il permet également l'analyse de code source lorsque couplé à PHPUnit (par exemple), afin de détecter le code mort. Un must-have.
Xdebug s'installe via PECL ou son site officiel (légèrement plus à jour). Voyez ce lien pour l'installation et le manuel des fonctions : Lien64.

### 7.17. docblock

ext/docblock est un parseur de commentaires au format PHPDoc (/** @foo=bar */). Très pratique lors de l'utilisation de la programmation orientée objets dans ses programmes. Ce parseur a maintes fois été réimplémenté en PHP (pour créer la documentation d'une classe de manière automatique par exemple), ici il s'agit d'une extension C, donc beaucoup plus performante.
Le manuel de ext/docblock est sur PECL (Lien65).

## 8. Conclusion

Voilà, j'espère vous avoir fait prendre conscience que compiler son logiciel manuellement, c'est avant tout en maîtriser les rouages et avoir la possibilité de personnaliser très hautement le programme en question. Dans le cas de PHP, qui est tout de même très modulaire, il y a du travail !
Plus d'excuses du type "oui mais je ne peux pas avoir cette extension" : vous avez ici toutes les billes pour compiler et créer votre propre PHP et vous avez déjà eu un aperçu de quelques extensions PECL sympathiques. Profitez de tous les atouts du monde open source !

Pour ceux qui ne sont pas du tout habitués à la compilation de sources sous Linux, je vous conseille grandement de vous y mettre si vous voulez creuser certaines options du *script configure*.
Lumière sur la compilation de programmes C : compilateur, assembleur, éditeur de liens et chargeur (Lien66).
Création de Makefiles : Lien67
Editeurs de liens et chargeurs : Lien68
Article dvp : La compilation séparée : Lien69

*Retrouvez l'article de Julien Pauli en ligne : Lien70*

## Comment créer facilement un framework Javascript - Partie 3

Traduction de l'article How to Easily Create a JavaScript Framework, Part 3 (Lien71) de Teylor Feliz paru sur AdmixWeb.

### 1. Introduction

Dans " Comment créer facilement un framework JavaScript - Partie 2 " (Lien72), j'ai parlé de quelques méthodes de navigation DOM, comme les techniques "getByName", "getByTag" et "getByClass". J'ai aussi expliqué comment créer d'autres méthodes supplémentaires juste pour le fun, telles que les fonctions "even" (pair), "odd" (impair) et "SetOpacity". Dans le tutoriel de cette semaine, je vais continuer cette série d'articles sur le framework JavaScript "VOZ" et continuer à ajouter de nouvelles méthodes au code de la partie 2. Je vais donc compléter un peu le framework en ajoutant des méthodes pour utiliser le JavaScript avant qu'une page ne soit chargée et d'autres techniques toutes aussi utiles. Continuez à lire la suite pour suivre l'évolution du framework ! N'hésitez pas à laisser vos commentaires et amusez-vous !

### 2. Attendre jusqu'au chargement du DOM

Avant de continuer avec notre framework, nous devons parler du chargement des pages Web. L'un des plus gros problèmes auquel un développeur peut être confronté est l'utilisation du JavaScript avant que la page ne soit complètement chargée. Dans de nombreuses applications web, le JavaScript entre en action après le chargement d'une page. C'est un gros problème car l'application ne fait rien jusqu'à ce que toutes les images et fichiers rattachés soient entièrement chargés. Pour expliquer ce phénomène, je vais utiliser un exemple.
Imaginons que nous ayons une page avec 20 images en miniatures pour présenter les produits d'une société. Imaginons que l'on utilise le JavaScript pour créer une popup sur chaque miniature pour que, lorsque l'on clique sur l'une d'elles, l'image apparaisse en taille réelle. Le problème est que l'application ne marchera pas tant que tous les éléments ne seront pas chargés. Notre page peut ressembler à ça :

```
function popUpImagen(imagen){
... ici le code ...
}
window.onload = popUpImagen;
```

Ce script attend que la page soit entièrement chargée pour commencer à lier les images en taille réelle aux miniatures. Si l'utilisateur à une connexion Internet lente, le script ne fonctionnera pas, tant que les tous les éléments de la page ne seront pas chargés.
Pour résoudre ce problème nous pouvons créer une petite méthode pour vérifier la disponibilité du DOM. Dès que celui-ci est chargé, nous pouvons travailler avec.

```
// Vérification du chargement du DOM
// Callback est une fonction anonyme qui
s'exécute dès que le DOM est prêt
domReady:function(callback){
  var done = false; // Création d'une variable
appelée done initialisée à false
  // Vérifie toutes les 10 millisecondes si le
document.body et le document.getElementById sont
chargés
  // S'ils sont disponibles, alors on change la
valeur de done en true
  var checkLoaded = setInterval(function()
{ if(document.body && document.getElementById)
{done = true;}},10);
  // Vérifie toutes les 10 millisecondes si done
== true
  // Si c'est le cas, alors on exécute la
fonction callback
  var checkInter = setInterval(function()
{if(done)
{clearInterval(checkLoaded);clearInterval(checkIn
ter);callback();}},10);
},
```

Utilisez donc cette méthode pour exécuter une partie du code aussitôt que le DOM est disponible. Exemple :

```
$$.domReady(function(){
      alert('Hello World');
});
```

### 3. Modification de la fonction "setStyle()"

Maintenant, modifions la méthode setStyle() pour utiliser un objet en paramètre. Il est souvent difficile de se souvenir de l'ordre des paramètres d'une fonction. Utiliser un objet est très pratique, car nous n'avons plus à nous préoccuper de l'ordre des paramètres. De plus, ce sera plus simple pour organiser ce que nous faisons.
Exemple d'une fonction qui insère des données en AJAX.

```
// Utilisation de paramètres littéraux
function
insertCustomer(prenom,nom,adresse,anniversaire,ge
nre,telephone){
... ici le code ...
}
```

Cela pourrait devenir pénible d'appeler cette fonction si nous ne nous souvenons pas de l'ordre des paramètres. Par exemple, nous pourrions utiliser la fonction comme ça :

```
insertCustomer('Teylor','Feliz','Mon
adresse','Male','09/15/1978','555-555-5555');
```

Nous recevons immédiatement une erreur, car nous

envoyons le paramètre "genre" à la place de "anniversaire". Une excellente façon d'éviter ce problème est de créer un objet et de le passer en paramètre de la fonction. De cette manière, nous devons juste connaître les noms des paramètres, plutôt que leur ordre d'envoi.
Exemple de l'appel de la même fonction lorsque l'on utilise un objet en paramètre :

```
insertCustomer({prenom:'Teylor', nom:'Feliz',
genre:'Male', telephone:'555-555-5555',
anniversaire:'09/15/1978', adresse:'Mon
adresse'});
```

Pour utiliser cet objet dans la fonction, nous devons la modifier comme dans l'exemple suivant :

```
function insertClient(args){
// Un exemple d'utilisation de la fonction est
args.nom
var prenom = args.prenom;
var nom = args.nom;
... ici le code ...
}
```

Revenons sur la méthode "setStyle" et à présent modifions-la pour qu'elle prenne en paramètre un objet.

```
// Mise à jour de la méthode setStyle
// Maintenant le paramètre est un objet
// Utilisation : var mystyle
={color:'red',background:'black'};
//     .setStyle(mystyle) ou alors nous pouvons
passer l'objet directement lors de l'appel
//
.setStyle({color:'red',background:'black'});
// Notez que le nom des styles utilisés sont les
mêmes qu'en JavaScript.
// Par exemple : background-color en css donne
backgroundColor en JavaScript, margin-left en css
donne marginLeft en JavaScript, etc.
setStyle:function(objStyle){
  for(var i = 0;i < this.elems.length;i++){//
Parcours des éléments et ajout des styles
      for(var k in objStyle){ // Parcours de
l'Objet ObjStyle en utilisant la propriété nom
comme nom de style
                              // et sa valeur
comme valeur de style
          // Par exemple : {top:20px} donnera
elem.style.top = 20px;
          this.elems[i].style[k] = objStyle[k];
      }
  }
  return this; // Renvoie this dans l'ordre
d'appel
},
```

Cela aide à envoyer plus d'une propriété de style en même temps. Exemple :

```
$$.getById('myid').setStyle({color:'red',
border:'1px solid blue'});
```

## 4. Méthodes set et get pour les éléments d'entrée

Les éléments d'entrée représentent la partie la plus importante d'une page Internet pour un développeur Web, car c'est le seul moyen d'obtenir des informations de l'utilisateur pour interagir avec ceux-ci en arrière-plan. C'est pourquoi nous allons créer deux fonctions : l'une pour afficher les données dans les éléments de saisie et l'autre pour récupérer ces mêmes données.

```
// Renvoie la valeur d'un élément ou un tableau
avec toutes les valeurs trouvées
// Cette fonction ne se chaine pas
getValue:function(){
  var elemstemp = [];// Création d'un tableau
temporaire pour sauvegarder les éléments trouvés
  for(var i=0;i < this.elems.length;i++){// On
parcourt tous les éléments pour vérifier leur
valeur
     if(this.elems[i].getAttribute('type') ==
'checkbox' || this.elems[i].getAttribute('type')
== 'radio'){
           // Si l'élément du formulaire est un
checkbox ou un bouton radio, on vérifie s'il est
coché
           // Si c'est le cas, on enregistre la
valeur
            if(this.elems[i].checked === true){
            elemstemp.push(this.elems[i].value);
            }
     } else{
            elemstemp.push(this.elems[i].value);
     }

   }
     //
elemstemp.push(if(this.elems[i].getAttribute)this
.elems[i].value); // Ajout de la valeur dans le
tableau temporaire
  if(elemstemp.length === 1){ // Si le tableau
temporaire a un seul élément, on renvoie
uniquement cet élément
     return elemstemp[0];
  }
  return elemstemp; // Sinon, on renvoie le
tableau temporaire avec toutes les valeurs
trouvées
},

// Mise à jour de la valeur pour les éléments
trouvés
// Val est la valeur pour les éléments trouvés
setValue:function(val){
  for(var i=0;i < this.elems.length;i++){ // On
parcourt tous les éléments pour vérifier leur
valeur
      // Si l'élément du formulaire est un
checkbox ou un bouton radio, on vérifie si val ==
elem.value
      // Si c'est le cas, alors on coche
l'élément
       if(this.elems[i].getAttribute('type') ==
'checkbox' || this.elems[i].getAttribute('type')
== 'radio'){
            if(this.elems[i].value === val){
              this.elems[i].checked = true;
            }
  }
  else{
       this.elems[i].value = val; // Mise à jour
de la valeur de l'élément
  }
  }
  return this;// Renvoie this dans l'ordre
d'appel
},
```

## 5. Effets amusants : "fadeIn()" et "fadeOut()"

On s'attend à trouver dans un Framework JavaScript de

superbes effets comme en DHTML, nous allons donc créer des méthodes pour faire apparaître ou disparaître des éléments en fondu. Mais, avant ça, nous devons créer un objet d'aide (helper) appelé "Helpers". L'objet d'aide va contenir les méthodes clés de nos fonctions d'effets. On ajoutera en plus quelques méthodes AJAX dont nous aurons besoin dans le chapitre suivant. Le code est bien commenté, je ne m'étendrai donc pas plus sur le sujet. La méthode d'aide est :

```
var Helpers={
// Méthode pour définir l'opacité d'un élément
// C'est presque le même que la méthode setOpacity()
// elem est l'élément sur lequel on va modifier l'opacité
// Niveau d'opacité souhaité, en pourcentage : les valeurs sont entre 0 et 100
setOpa:function(elem,level){
                if(level>=0 && level< =100){// La valeur de l'opacité doit être entre 0 et 100
                    elem.style.opacity = (level/100); // On définit l'opacité  pour Firefox, Safari, Opera, Chrome,etc.
                    elem.style.filter = 'alpha(opacity='+level+')'; // On définit l'opacity pour IE :(
                }
            },
// Méthode pour créer l'effet de disparition avec fondu
// Utilisation de la méthode setOpa
// elem est l'élément sur lequel on va modifier l'opacité
// time est la durée de l'effet
fadeOut:function(elem,time){
  var level = 100; // Définit le niveau à 100
  var interval = setInterval(function(){ // On crée un intervalle en utilisant la méthode setOpa
       Helpers.setOpa(elem,--level); // On utilise la méthode setOpa pour décrémenter le niveau de 1
       if(level==0){ // Si le niveau a pour valeur 0, alors on stoppe l'intervalle
          clearInterval(interval);
       }
  },time/100);
},

// Méthode pour créer l'effet d'apparition avec fondu
// Utilisation de la méthode setOpa
// elem est l'élément sur lequel on va modifier l'opacité
// time est la durée de l'effet
fadeIn:function(elem,time){
  var level = 0;// Définit le niveau à 0
  var interval = setInterval(function(){// On crée un intervalle en utilisant la méthode setOpa
       Helpers.setOpa(elem,++level);  // On utilise la méthode setOpa pour incrémenter le niveau de 1
       if(level==100){// Si le niveau a pour valeur 100, alors on stoppe l'intervalle
          clearInterval(interval);
       }
  },time/100);
}

}
```

Ajoutons maintenant les méthodes "fadeIn" et "fadeOut" dans le framework "VOZ".

```
// Méthode de disparition avec fondu
// Utilisation de l'objet Helpers
// time est en millisecondes par exemple 8 secondes = 8000
fadeOut:function(time){
  for(var i=0; i < this.elems.length;i++){// Parcours des éléments et exécution de la fonction fadeOut du helpers
          Helpers.fadeOut(this.elems[i],time)
  }
  return this;// Renvoie this dans l'ordre d'appel
},

// Méthode d'apparition avec fondu
// Utilisation de l'objet Helpers
// time est en millisecondes par exemple 8 secondes = 8000
fadeIn:function(time){
  for(var i=0; i < this.elems.length;i++){
          Helpers.fadeIn(this.elems[i],time)
  }
  return this;// Renvoie this dans l'ordre d'appel
},
```

## 6. Conclusion

Exemple du framework JavaScript VOZ partie 3 : Lien73

Comme dans les parties 1 et 2, j'ai fait une page qui vous montre en action la partie 3 du framework JavaScript "VOZ", visitez le lien ci-dessus. Aussi, dans la quatrième partie de cette série sur le framework JavaScript VOZ, nous créerons d'autres méthodes AJAX et je vous montrerai comment les utiliser. Comme toujours, j'espère que vous avez trouvé ce tutoriel amusant et facile à suivre ! N'hésitez pas à laisser vos commentaires.

## 7. Articles de cette série

- Comment créer facilement un framework Javascript - Partie 1 : Lien74
- Comment créer facilement un framework Javascript - Partie 2 : Lien72
- Comment créer facilement un framework Javascript - Partie 3 : Lien75

*Retrouvez l'article de Teylor Feliz traduit par KalyParker en ligne : Lien75*

## Interview de James Reinders

Intel propose des outils de développement orientés développement parallèle pouvant tirer parti des architectures matérielles actuelles. Dans cette interview, James Reinders, le gourou d'Intel sur le sujet et auteur d'un livre sur les Thread Building Blocks, fait le point sur les outils prévus en 2010.
J'avais déjà réalisé une interview de James l'année dernière (Lien76), à laquelle il peut être intéressant de se référer pour des questions plus générales. En outre, j'ai rédigé un compte rendu de la conférence en elle-même (Lien77).

### 1. Questions générales

**Pendant la conférence, vous avez mentionné à de multiples reprises le compilateur C++ d'Intel et celui de Microsoft. Qu'en est-il du compilateur Clang, sponsorisé par Apple ?**

Nous avons discuté avec eux et sommes très au courant de ce qu'ils font. Notre compilateur Fortran et C++ fonctionne avec Mac Os X et nous comptons continuer ainsi. Beaucoup de gens apprécient que nous supportions plusieurs plateformes, c'est une des principales raisons de la popularité de nos compilateurs.

Nous nous échangeons des plans, et parfois des technologies ou des revues avec Apple. J'ai assisté à une présentation chez Intel il y a environ deux ans présentant comment ils s'éloignaient de GCC et présentant la technologie LLVM (sur laquelle est construit Clang). Mais nous n'investissons pas de manière massive dans leur technologie de compilateur.

**Pourquoi CILK est présenté à l'utilisateur, et non pas simplement utilisé en interne par des bibliothèques comme TBB.**

L'objectif de CILK est de prendre un programme fonctionnel et de le rendre parallèle avec le minimum de modifications, comme changer un for en cilk_for, un appel de fonction par un cilk_spawn. Son extrême simplicité devrait en faire un point d'accès aisé au monde du parallélisme. Sa syntaxe est faite de telle façon, qu'en ignorant les mots clefs spéciaux, on se retrouve avec le code initial, ce qui rend la compréhension facile.

### 2. Parallel Studio

**Quand la prochaine version de Parallel Studio, compatible avec Visual Studio 2010, est-elle prévue ?**

Le programme de bêta officiel devrait commencer vers mi-mai. La sortie est prévue pour la fin du troisième trimestre, nous espérons pouvoir le sortir pour l'IDF (Intel Developper Forum (Lien78)), à la mi-septembre, à San Francisco.

**Quelles seront les nouvelles fonctionnalités ?**

Tout d'abord, comme indiqué, le support pour Visual Studio 2010. Nous avons dû attendre que ce produit soit finalisé.

Le compilateur prendra aussi en compte Cilk, avec l'ajout de trois primitives, ainsi que la notation tableau. Cette notation sera comprise par le C et le C++. Voici quelques exemples :

```
a[:] = b[:] + 3; // Permet de copier les éléments d'un tableau à une dimension vers un autre
```

```
a[:] = b[3][:] * c[:][12]; // Extrait des vecteurs lignes ou colonnes pour faire des calculs
```

Cette notation, semblable à celle de Matlab, pourra automatiquement bénéficier d'une parallélisation, de SSE, d'AVX. Nous travaillons avec d'autres constructeurs de compilateurs, pour se mettre d'accord sur une syntaxe, et peut être au final, basé sur l'expérience apportée par une implémentation dans plusieurs compilateurs, en tirer une proposition pour les comités de normalisation C et C++.

Intel Parallel Advisor a été en phase d'incubation cette année, mais deviendra un produit à part entière dans cette version. Il permet d'accélérer une étape critique de développement parallèle. Quand on se lance dans la parallélisation, la première étape est d'essayer de voir où l'on souhaite paralléliser, la seconde est d'écrire le code correspondant et d'effectuer des mesures.

Mais cette seconde étape peut être longue, des semaines ou des mois pour trouver les bugs, les race conditions, les deadlocks... pour parfois au final se rendre compte que ça ne marche pas aussi bien qu'on l'avait espéré. Nous essayons d'accélérer cette étape les développeurs ajoutent dans leur code des annotations indiquant leur souhait d'en paralléliser une portion, sans vraiment coder cette parallélisation. Un outil prend alors le relais pour analyser ce code et indiquer : si vous faites ça, vous allez avoir des race conditions, ou bien votre code ne sera pas scalable.

Ces notations permettent ainsi de prototyper rapidement le code, il n'est même pas nécessaire de l'exécuter sur une machine multi-cœurs. Ce produit est conçu pour aider les architectes à choisir entre différentes manières de paralléliser du code, avant d'investir dans la parallélisation elle-même.

**Dans Visual Studio 2010, Microsoft a introduit la**

**notion de tâche (par opposition aux threads) dans le débogueur. Avez-vous une notion semblable ?**

Nous avons ajouté des annotations dans TBB afin que notre Inspector (qui détecte des race conditions) ou Amplifier (qui mesure des performances) comprenne la notion de tâche au sens TBB. Microsoft a fait de même avec les tâches de TPL et PPL afin que leur propre débogueur puisse les comprendre. Mais ces annotations ne sont pas compatibles. Nous étudions ce que nous pouvons faire afin d'harmoniser la situation, que nos outils comprennent les tâches de TPL et PPL et que leurs outils comprennent les tâches de TBB.

Les utilisateurs veulent pouvoir utiliser un outil unique dans tous les cas. Nous avons déjà écrit des extensions à leur débogueur qui améliorent son comportement avec OpenMP, ils ont ajouté des extensions très agréables pour les tâches, nous regardons si on peut communiquer avec leur débogueur afin d'en tirer les même bénéfices avec TBB.

**Une partie de Parallel Studio devrait être disponible sous Linux. Quand ? Quel ordre de grandeur de prix ?**

Novembre. Le programme bêta devrait commencer en juin/juillet, pour une sortie du produit en novembre.

En terme de prix, on devrait être dans la même gamme que nos autres outils de développement (compilateur, VTune), et donc un peu plus cher que Parallel Studio. Le compilateur Fortran sera en option, ce qui pourrait augmenter le prix. Mais nous n'avons pas encore annoncé de tarif actuellement.

## 3. TBB

**Les styles de programmation entre TBB et PPL sont très proches, pensez-vous qu'il est possible, en se restreignant à un sous-ensemble de ces bibliothèques, d'avoir un code source qui serait compatible avec ces deux bibliothèques ?**

Il est vrai que ces bibliothèques sont très semblables dans leur interface. On pourrait imaginer un programme qui juste avec un peu de macros permettrait de basculer de l'un à l'autre. Ce qui est important, c'est que la sémantique est la même. Par exemple, il se passe la même chose quand on utilise le parallel_for de TBB et celui de PPL. La syntaxe peut légèrement varier, puisque par exemple ils gèrent les indices légèrement différemment de nous.

Pour les structures de données (concurrent_queue, concurrent_vector...), c'est encore plus vrai, puisqu'il s'agit du même code, Microsoft ayant repris le code d'Intel. Il n'y avait en effet aucune raison de ne pas avoir les mêmes structures de données.

**Quelles sont les nouveautés de la version 3 de TBB ?**

La version 3 va être officiellement annoncée la semaine prochaine. Tout d'abord, sous Windows, il y a l'utilisation du concurrency runtime, que Microsoft vient juste de sortir. Il y a aussi plein de modifications que je qualifierais d'entretien. Par exemple, on a effectué un certain nombre d'ajustements pour la prise en compte de C++0x. On gérait déjà les lambda-fonctions, par exemple dans un parallel_for, mais pas dans l'ensemble de TBB, en particulier pour la gestion des pipelines.

Un autre exemple est la manière dont fonctionne le thread principal, qui a lancé les autres threads utilisés par TBB. En général, ce thread fait tourner des tâches, comme les autres threads, et ça ne pose pas de problème. Mais selon la manière dont on prend des verrous, ce comportement peut produire des deadlocks et la nouvelle version permet de régler ce comportement, et de rendre le thread principal passif.

Ce que les utilisateurs de TBB apprécient beaucoup, c'est qu'elle est suffisamment mature pour être mise en œuvre, et résoudre de vrais problèmes dans de vrais programmes.

## 4. Ct

**Est-ce que Ct sera distribué en open source ?**

Pour l'instant, on ne sait pas vraiment quelle est la bonne façon de faire. Si TBB a été placée en open source, c'est parce que c'était la bonne manière de la rendre disponible sur de nombreuses plateformes. OpenMP présente un autre modèle, une API standardisée que plusieurs personnes peuvent implémenter.

Nous voulons que Ct soit disponible sur de nombreuses plateformes, et nous sommes en discussions avec des compagnies qui pourraient en réaliser un portage. Nous ne savons pas encore ce qui serait le mieux, entre une solution open source et une solution basée sur une API ouverte et des implémentations propriétaires tirant parti des bénéfices de chaque plateforme. Il est en tout cas probable que Ct ne soit pas uniquement une technologie propriétaire.

Nous voulons aussi avoir une API ouverte au niveau du front-end, afin de permettre l'interface avec d'autres langages, comme par exemple Python ou Perl.

**Quels compilateurs seront supportés pour la première version ?**

Au minimum, Intel Compiler, Microsoft Visual C++ et gcc. Mais la technologie n'est pas très dépendante du compilateur et devrait marcher avec n'importe quel compilateur C++. On avait eu des problèmes avec TBB, en particulier sur la gestion des exceptions, qui nous avait forcés à faire des modifications, par exemple pour le compilateur Solaris. C'était il y a longtemps, mais c'est pourquoi je ne m'avance pas plus pour Ct.

**Quels sont les concepts techniques derrière Ct. Est-ce plus qu'une bibliothèque de matrice gérant la parallélisation ?**

Ct crée le concept d'un environnement protégé, avec des variables sur lesquelles on ne peut pas tout faire (par exemple, prendre des pointeur dessus). Ça nous donne un contrôle complet sur ces variables, et on peut donc prendre en compte des processeurs spécialisés, de la mémoire distribuée... Le problème des variables en C et C++ est

qu'il est possible de prendre leur adresse de tant de manières différentes, qu'on ne peut pas bouger les données en mémoire sans casser le programme.

Avec Ct, tant qu'on ne prend pas de pointeur sur une variable, on a donc une latitude d'optimisation supplémentaire, qui nous permet d'utiliser du GPGPU, l'architecture Larrabee, ou le CPU. Pouvoir bouger la donnée est donc l'un des concepts principaux de Ct.

**La prise en compte de toutes ces architectures aura-t-elle lieu à la compilation ou à l'exécution ?**

A l'exécution. Nous avons une technologie de JIT. Par exemple, quand on exécute le code, si l'on détecte la présence d'un GPU, on peut décider de lui déléguer tout un problème mathématique, ou encore de partager cette tâche entre un CPU et le GPU.

C'est un aspect important du design, ce n'est pas une simple bibliothèque de templates générant directement du code, elle génère des opérations, qui seront ensuite envoyées vers un système gérant leur exécution. Ce système pourra décider de bouger les données, d'agréger ou pas des opérations, de les distribuer sur diverses unités de calcul...

Ce qui est intéressant, c'est que le code source reste simple, alors que tant de choses complexes sont utilisées pour le mettre en œuvre. Nous pensons que cette technologie de JIT va être très efficace, mais elle est complexe et nous devons faire attention à pleins de détails.

**Remarque : James doit m'envoyer des papiers entrant un peu plus dans les détails. Peut-être un prochain article ? Une présentation plus détaillée est aussi prévue pendant le salon SuperComputing aux US en novembre.**

**Qu'est-ce qui va changer dans Ct suite à l'acquisition de RapidMind ?**

Principalement deux points :

Tout d'abord, ils avaient de très bonnes technologies pour cibler la GPU, nous allons en bénéficier. Ils nous permettent aussi de rendre plus prévisible notre technique de JIT.

Ensuite, ils ont de nombreux retours utilisateurs sur leur technologie et ont pu proposer des modifications de syntaxe, comment la rendre plus en phase avec le C++ et plus simple. Par exemple, dans certains cas, du code écrit avec la syntaxe actuelle prend trois ou quatre fois plus de place que du code écrit avec la prochaine version.

*Retrouvez l'article de Loïc Joly en ligne : Lien79*

## Visual C++ 2010 premières impressions

Depuis quelques mois je travaille avec Visual Studio 2010 dans un premier temps avec les bêta 1 et 2 et maintenant avec la RC disponible depuis les TechDays 2010.
Mon environnement de travail est un PC sous Seven 64 bits avec deux écrans, 2 Go de mémoire et un processeur Core Duo 6600 à 2,40 GHz.
L'installation n'a pas causé de problème particulier et cela fait presque deux mois que je travaille sur des projets migrés de Visual Studio 2008.

### 1. Comment se comporte cette version en termes de rapidité ?

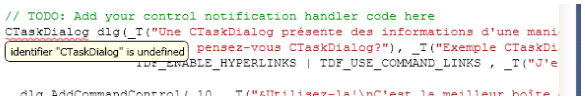
Je n'ai pas constaté de problèmes de lenteur, cette version se comporte normalement.

### 2. Vraiment finie la galère du fichier .ncb et de la perte de l'IntelliSense ?

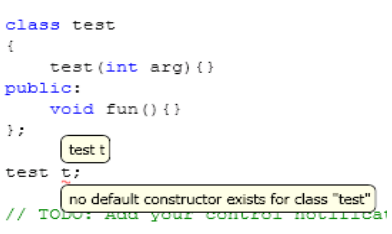
Comme vous l'avez peut être appris, cette nouvelle version ne travaille plus avec le fichier .ncb mais utilise une base SQL (locale) pour tenir à jour les informations relatives aux classes (classview) et d'IntelliSense.
En cas de problème nous étions habitués à supprimer le .ncb, dorénavant une nouvelle commande "Relancer l'analyse de la solution", apparaît dans le menu projet pour scanner la solution.
Le fichier .ncb porte l'extension .sdf et est exploité par SQL Server Compact database.
Un onglet de paramétrage de l'IntelliSense a été rajouté dans les options à l'emplacement : Menu Outils/Options/Editeur de texte/ C/C++ /Avancé.

Une section qui peut être intéressante à modifier est " Emplacement de secours ".

Par défaut l'IDE utilise le répertoire de votre solution pour stocker les fichiers .sdf.
En indiquant un répertoire dans la rubrique " Emplacement de secours " l'IDE utilisera cet emplacement pour créer ces fichiers.
N'oubliez pas de paramétrer votre anti-virus pour exclure ce fichier de l'analyse.
IntelliSense a aussi évolué puisqu'il donne maintenant des informations sur la validité ou la cohérence du code écrit,
**Exemples :**




**Ou encore :**



À l'usage cette fonctionnalité est très pratique, d'un simple coup d'œil je sais que mon code ne compilera pas, alors que précédemment il faillait lancer une compilation pour voir les erreurs…

### 3. Puis-je distribuer mon programme MFC ou C++ avec les DLL systèmes associées ?

La réponse est oui, il est maintenant possible de distribuer son programme dans un répertoire avec les DLL MFC et celle de la CRT.
Il ne sera donc plus nécessaire à priori d'utiliser vcredist.

### 4. Quelles sont les nouveautés de l'éditeur et de l'interface ?

La nouvelle interface utilise WPF. Le premier apport important est le zoom vectoriel rendu possible sur un fichier source.
Il suffit d'utiliser la molette de la souris avec la touche CTRL enfoncée pour zoomer sur le texte.

**- Gestion des projets récents :**

Les projets ouverts peuvent être punaisés comme sur la barre d'outils de Seven.

**- Nouvelles possibilités de l'interface MDI :**

Le menu contextuel sur l'onglet du document donne de nouvelles fonctionnalités :

Une fonctionnalité qui me manquait depuis longtemps : la commande " flotter " qui permet de détacher la fenêtre des onglets MDI, cela permet de disposer un élément en dehors de l'interface de l'éditeur.
Cette fonctionnalité s'applique bien sûr aux autres éléments de l'interface comme le gestionnaire de projets ou par exemple un fichier source.
Les fenêtres de l'interface se ferment par la croix directement disponible sur l'onglet.

**- Personnaliser la page d'accueil de Visual Studio :**

Quand j'ai commencé à m'intéresser à ce sujet sur les versions bêta il fallait travailler sur un fichier au format XAML, avec la RC exit le format XAML retour au XML

Les fichiers à modifier sont situés à l'emplacement d'installation de Visual Studio 2010, ce qui donne sur ma machine :
D:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\IDE\StartPages\en
Il y a un fichier par type d'environnement, dans mon cas j'ai modifié le fichier Links.Ultimate.VC.xml.
Rajouter une nouvelle section est un jeu d'enfant. Je me suis concocté une nouvelle section pour rester connecté avec ma tribu :

J'ai commencé par capturer la page d'accueil de developpez.com que j'ai enregistrée dans un fichier au format .png
J'ai ensuite rajouté la section suivante juste en dessous de celle de " Community and Learning Resources "

```
<item>
  <title id="welcome_developpez_title">Communauté developpez.com</title>
  <image>D:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\IDE\StartPages\en\developpez.PNG</image>
  <description id="welcome_developpez_desc">le club des professionnels de l'informatique</description>
  <commands>
    <command>
      <commandtype>Browse</commandtype>
      <title id="welcome_developpez_cmd1_title">La Faq Visual C++</title>
      <parameter>http://cpp.developpez.com/faq/vc/</parameter>
    </command>
    <command>
      <commandtype>Browse</commandtype>
      <title id="welcome_developpez_cmd2_title">La Faq C++</title>
      <parameter>http://cpp.developpez.com/faq/cpp/</parameter>
    </command>
    <command>
      <commandtype>Browse</commandtype>
      <title id="welcome_developpez_cmd3_title">Forums Visual C++</title>
      <parameter>http://www.developpez.net/forums/f29/c-cpp/outils-c-cpp/visual-cpp/</parameter>
    </command>
  </commands>
</item>
```

À mon avis la solution retenue en XML est plus simple que la précédente en XAML.

**- Cibler une autre plateforme de construction du projet :**

Il est possible de spécifier Visual Studio 9.0 pour la construction du binaire de votre application, dans ce contexte, Visual Studio 2010 utilisera Visual Studio 2008 (qui devra être présent sur la machine) pour construire le projet. Ce réglage se fait directement dans votre projet à l'emplacement suivant: Propriétés de configuration/Ensemble d'outils de plateforme.

### 5. Gestion des chemins de recherche

Avant vous définissiez vos différents chemins de recherche pour les sources, bibliothèques etc dans le menu options de Visual Studio.
Eh bien, c'est fini.
Ce paramétrage a migré dans votre projet !

Vous pouvez exporter le paramétrage des chemins de recherche définis globalement dans Visual studio 2008 pour les réintégrer dans Visual Studio 2010.
Au final, dans chaque nouveau projet ou projet migré, on retrouve l'ensemble des chemins de recherche dans les différentes sections.
Pour définir globalement vos chemins de recherche, il existe plusieurs solutions : pour les définir par l'IDE, il faut appeler à partir d'un projet l'option affichage / Gestionnaire de propriétés.

L'item **Microsoft.Cpp.Win32.user.props** contient la

définition des chemins par défaut.
Il est possible de modifier directement ce fichier au format xml, stocké par l'utilisateur.
Il est situé à l'emplacement (sous seven) :
**C:\Users\user\AppData\Local\Microsoft\MSBuild\v4.0**
En voici un exemple vierge de toutes modifications:

```
<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build"
ToolsVersion="4.0"
xmlns="http://schemas.microsoft.com/developer/msb
uild/2003">
  <PropertyGroup>
    <ExecutablePath>$
(ExecutablePath)</ExecutablePath>
    <IncludePath>;$(IncludePath)</IncludePath>
    <ReferencePath>$
(ReferencePath)</ReferencePath>
    <LibraryPath> $(LibraryPath)</LibraryPath>
    <SourcePath>$(SourcePath)</SourcePath>
    <ExcludePath>$(ExcludePath)</ExcludePath>
  </PropertyGroup>
</Project>
```

Il vous faudra donc rajouter toutes vos définitions globales dans ce fichier en séparant les chemins par des " ; ", ou en passant par l'interface de l'IDE.

Bien qu'on arrive au même fonctionnement qu'avec Visual Studio 2008, la différence tient dans le paramétrage, qui a migré dans un fichier stocké par l'utilisateur.

## 6. Quelles sont les nouveautés sur les MFC ?

**- La classe CTaskDialog :**

Cette nouvelle classe permet de définir rapidement des boîtes à messages plus évoluées que la classique boîte de dialogue obtenue par la fonction MessageBox.
Elle ne sera disponible qu'avec Windows Vista et Seven, et est une encapsulation directe de l'API Win32.
La méthode DoModal appelle directement dans COMCTL32.DLL la fonction TaskDialogIndirect.

Un petit exemple ?
J'ai rajouté ces quelques lignes dans mon projet de tests sur un bouton d'une FormView :

```
void Ctestvc2010View::OnBnClickedButtontaskdlg()
{
// TODO: Add your control notification handler
code here
CTaskDialog dlg(_T("Une CTaskDialog présente des
informations d'une manière claire et cohérente"),
_T("Que pensez-vous de CTaskDialog?"),
_T("Exemple CTaskDialog"), 0,
                    TDF_ENABLE_HYPERLINKS |
TDF_USE_COMMAND_LINKS , _T("J'espère que vous
l'aimez!"));

      dlg.AddCommandControl( 10, _T("&Utilisez-
la!\nC'est la meilleure boîte dialogue que vous
pouvez avoir!!"));
      dlg.AddCommandControl( 20, _T("&Peut-
être?\nVous ne voulez pas l'essayer ?"));
      dlg.AddCommandControl( 25, _T("&Aucune
chance!\nJe ne vais pas l'utiliser!"));

      dlg.AddRadioButton( 3, _T("Beaucoup"));
      dlg.AddRadioButton( 7, _T("Un Petit peu
"));
      dlg.AddRadioButton( 4, _T("Pas du tout"));

      dlg.SetMainIcon(TD_SHIELD_ICON);
      dlg.SetFooterIcon(TD_INFORMATION_ICON);
      INT_PTR nResult = dlg.DoModal();

}
```

C'est une traduction directe de l'exemple que l'on peut trouver sur le blog de l'équipe Visual US.
**- Redémarrage de l'application après un crash :**

Ce système que vous avez peut être déjà expérimenté dans Office 2007 ou Visual Studio permet de relancer l'application avec une restauration de l'environnement de travail, qui est régulièrement sauvegardé par l'application.
La bibliothèque MFC implémente donc maintenant ce mécanisme qui devient une option de l'assistant à la génération du programme.

**- DPI Awareness :**

Cette option permet d'adapter le texte et les contrôles de nos applications MFC en fonction du réglage des fontes Windows : la petite fonte correspondant à 96 dpi, la moyenne à 120 dpi et la grande à 144 dpi.
Cette option est activable dans les options du projet à l'emplacement : Outil Manifeste/Prise en charge DPI.
**- Gestion des transactions sur les opérations fichier et sur les fonctions associées à la Base de registre:**
Une nouvelle classe ATL CAtlTransactionManager permettant de gérer les transactions des fichiers a été rajoutée.
Les différentes classes MFC de manipulation de fichiers ou de répertoires ont été modifiées pour supporter cette fonctionnalité.

```
CFile(CAtlTransactionManager *);
CStdioFile(CAtlTransactionManager *);
CAtlFile(CAtlTransactionManager *);
CFileFind(CAtlTransactionManager *);
```

Les API : AfxRegCreateKey, AfxRegOpenKey, AfxRegDeleteKey disposent désormais d'un paramètre TransactionManager.

Exemple avec AfxRegCreateKey :

```
LONG AFXAPI AfxRegCreateKey(HKEY hKey, LPCTSTR
lpSubKey, PHKEY phkResult,
CAtlTransactionManager* pTM = NULL);
```

Cet exemple issu du blog US de Visual Studio illustre l'utilisation de la transaction (commit) et du retour arrière (rollback) :

```
 CAtlTransactionManager tmCommit;
 CFile
cfile1(_T("c:\\TransactionFileSystemDemo\\cfile1.
txt"), CFile::modeCreate, &tmCommit);

 CStdioFile
stdiofile1(_T("c:\\TransactionFileSystemDemo\\cst
diofile1.txt"), CFile::modeCreate, &tmCommit);

 CAtlFile atlfile1(&tmCommit);
```

```
 atlfile1.Create(_T("c:\\TransactionFileSystemDem
o\\atlfile1.txt"), STANDARD_RIGHTS_REQUIRED,
FILE_SHARE_READ, CREATE_ALWAYS,
FILE_ATTRIBUTE_NORMAL);

 tmCommit.Commit();
 tmCommit.Close();

CAtlTransactionManager tmRollback;
CFile
cfile2(_T("c:\\TransactionFileSystemDemo\\cfile2.
txt"), CFile::modeCreate, &tmRollback);
 CStdioFile
stdiofile2(_T("c:\\TransactionFileSystemDemo\\cst
diofile2.txt"), CFile::modeCreate, &tmRollback);
 CAtlFile atlfile2(&tmRollback);

 atlfile2.Create(_T("c:\\TransactionFileSystemDem
o\\atlfile2.txt"), STANDARD_RIGHTS_REQUIRED,
FILE_SHARE_READ, CREATE_ALWAYS,
FILE_ATTRIBUTE_NORMAL);

 tmRollback.Rollback();
tmRollback.Close();
```

**- Le support du multi-touch :**

Les applications MFC peuvent prendre en compte le multi-touch de Windows Seven.
Voici rapidement comment procéder : dans la méthode InitInstance de votre classe d'application rajouter les lignes suivantes pour tester le support matériel du multi-touch :

```
BYTE digitizerStatus = (BYTE)
GetSystemMetrics(SM_DIGITIZER);
if ((digitizerStatus & (0x80 + 0x40)) == 0)
//Stack Ready + MultiTouch
{
 AfxMessageBox(L"Pas de périphérique Multi-Touch
disponible");
         return FALSE;
}
BYTE nInputs = (BYTE)
GetSystemMetrics(SM_MAXIMUMTOUCHES);
CString str;
str.Format(L"Périphérique Multi-Touch disponibl
avec  %d points.", nInputs);
//AfxMessageBox(str);
```

Pour indiquer que votre application cliente est apte a recevoir les messages Multi-Touch, il faudra appeler la méthode **CWnd::RegisterTouchWindow()** dans la méthode de réponse au message **WM_CREATE** de votre fenêtre fille (la child ) de traitement.

```
int CChildFrame::OnCreate(LPCREATESTRUCT
lpCreateStruct)
{
        if
(CMDIChildWndEx::OnCreate(lpCreateStruct) == -1)
                return -1;
        if (!RegisterTouchWindow()) return -1;
}
```

L'appel de cette méthode rend l'application capable de recevoir les messages **WM_TOUCH**.
Il vous faut ensuite rajouter manuellement dans la child la méthode suivante :

```
virtual BOOL OnTouchInput(CPoint pt, int
nInputNumber, int nInputsCount, PTOUCHINPUT
pInput);
```

Cette méthode va nous permettre de traiter les différents événements possibles :

```
BOOL CChildView::OnTouchInput(CPoint pt, int
nInputNumber, int nInputsCount, PTOUCHINPUT
pInput)
{
   if ((pInput->dwFlags & TOUCHEVENTF_DOWN) ==
TOUCHEVENTF_DOWN) // Touch Down event
   {
        // votre traitement
        return TRUE;
   }
   else if ((pInput->dwFlags & TOUCHEVENTF_MOVE)
== TOUCHEVENTF_MOVE) // Touch Move event
   {
        // votre traitement
        return TRUE;
   }
   else if ((pInput->dwFlags & TOUCHEVENTF_UP) ==
TOUCHEVENTF_UP) // Touch Move event
   {
        // votre traitement
        return TRUE;
   }

   return FALSE;
}
```

Voilà pour la mise en place de l'architecture.
Il ne reste plus qu'à mettre en place votre traitement.

**- Le retour de Class Wizard :**

Cet outil, présent dans Visual 6.0 a été enlevé avec Visual Studio 2002.
Il fait son grand retour dans Visual Studio 2010.
Il faut dire que cela fait des années que tout le monde le réclame.

Le raccourci clavier n'est pas le même que sous Visual 6.0 CTRL+X mais CTRL+MAJ+X
On retrouve enfin les mêmes fonctionnalités dont nous disposions sous Visual 6.0.

Pour ceux qui n'ont pas connu Class Wizard c'est une boîte à onglets permettant de réaliser l'ensemble des opérations possibles sur une classe fenêtre : messages, ajout de données membres, etc.
Plutôt que de courir à différents endroits pour effectuer des opérations, tout est disponible dans un seul outil.
D'autre part, différentes actions peuvent être réalisées en même temps, comme l'ajout de plusieurs variables associées à des contrôles, ce qui n'était pas (plus) le cas avant.

**- Image de fond dans les dialogues :**
Il est maintenant possible de rajouter une image en fond d'une boîte de dialogue avec la " mockup bar " , celle-ci est disponible en bas de l'éditeur de ressources.
Un clic sur le bouton de parcours de fichier dans la barre permet de sélectionner une image que l'on pourra positionner grâce aux zones d'offset dont on pourra régler

l'opacité.
Malgré cela je ne peux que regretter l'absence d'un dispositif permettant de régler la couleur de fond d'une boîte de dialogue ou d'une FormView.

**- Autorisation d'affichage des ActiveX dans l'éditeur de ressources :**
Si des ActiveX sont présents dans vos ressources, Visual Studio vous demandera l'autorisation de les afficher, ce qui risque à la longue de devenir fatiguant.

Certains d'entre vous ont dû remarquer depuis Visual Studio 2008 l'absence de l'application qui permettait de tester un Activex, celle-ci est en fait disponible dans les exemples des MFC situés à l'emplacement suivant :
**pour VS2008 : [Program Files]\Microsoft Visual Studio 9.0\Samples\1033\AllVCLanguageSamples.zip ;**
**pour VS2010: [Program Files]\Microsoft Visual Studio 10.0\Samples\1033\VC2010Samples.zip.**
L'exemple se nomme **TSTCON** dans le dossier **OLE**.

**- Prévisualisation des documents dans la barre de Windows Seven :**

J'ai généré une application multi-document MFC avec une CeditView.
Lorsque j'ouvre trois onglets dans mon application MFC ils deviennent visibles dans la barre d'application de Windows Seven.
De plus, le clic sur le document dans la barre des taches active l'onglet correspondant dans l'application.
Au moment de la génération de l'application, j'ai coché les options de prévisualisation du document et de l'affichage simplifié.

Ces options permettent la prévisualisation du document dans l'explorateur Windows, son affichage simplifié (Thumbnail) et la recherche de contenu.
Toute la logique de traitement est générée automatiquement dans la classe document.
Il faudra adapter ce code pour vos documents.

**- Programmation du Ruban (ribbon) :**

Le ruban est maintenant directement éditable dans l'éditeur de ressources.
Sa description est au format XML.
Sa composition est donc moins laborieuse que dans Visual Studio 2008 où il fallait tout coder manuellement.

L'ensemble des contrôles nécessaires à la création du ruban est donc disponible dans la Boîte à outils.
Une remarque personnelle : le ruban est beau mais il ne se prête pas à toutes les applications.
Les écrans d'aujourd'hui sont pratiquement tous au format large (wide) et la perte de surface en hauteur provoquée par ce style d'interface est très pénalisante.
Il serait intéressant (c'est un appel du pied) de disposer de la même interface à la verticale de l'écran…

## 7. Support partiel de C++0x

Le tableau ci-contre expose les avancées du support de C++0x pour Visual Studio 2010 par rapport à Visual Studio 2008 :

| C++0x : fonctionnalités de langage | Proposition | VC9 | VC10 |
|---|---|---|---|
| Références Rvalue | N2118 (Lien80) | NON | V2 |
| Références Rvalue v2 | N2844 (Lien81) | NON | V2 |
| Références Rvalue pour this | N2439 (Lien82) | NON | NON |
| Initialisation de classe d'objets par rvalues | N1610 (Lien83) | OUI | OUI |
| static_assert | N1720 (Lien84) | NON | OUI |
| auto | N1984 (Lien85) | NON | OUI |
| Déclarations multiples avec auto | N1737 (Lien86) | NON | OUI |
| Suppression de l'ancienne fonctionnalité auto | N2546 (Lien87) | NON | OUI |
| Mise en place du type de retour | N2541 (Lien88) | NON | OUI |
| Lambdas | N2927 (Lien89) | NON | V1.0 |
| decltype | N2343 (Lien90) | NON | OUI |
| Crochets en chevron à droite (>>) | N1757 (Lien91) | OUI | OUI |
| Modèle (template) Externe | N1987 (Lien92) | OUI | OUI |
| nullptr | N2431 (Lien93) | NON | OUI |
| enums Fortement typés | N2347 (Lien94) | Partiel | Partiel |
| Pré- déclaration des enums | N2764 (Lien95) | Partiel | Partiel |
| Déclarations amies (friend) étendues | N1791 (Lien96) | Partiel | Partiel |
| Type local et non nommé en tant qu'arguments de modèles | N2657 (Lien97) | OUI | OUI |
| Fonctionnalités de langage de base: Concurrence | | | |
| exception_ptr | N2179 (Lien98) | NON | OUI |
| Stockage de données local au thread (TLS) pour info cette technique est utilisée dans les MFC. | N2659 (Lien99) | Partiel | Partiel |
| Fonctionnalités de langage de base: C99 | | | |
| __func__ | N2340 (Lien100) | Partiel | Partiel |
| Préprocesseur C99 | N1653 (Lien101) | Partiel | Partiel |
| long long | N1811 (Lien102) | OUI | OUI |

## 8. La nouvelle bibliothèque sur le parallélisme en mode natif : Parallel pattern Library

Cette bibliothèque permet de paralléliser des applications natives en C++.
Elle rentre en concurrence avec celle d'Intel : TBB (voir mon compte rendu des Techdays 2008 (Lien103))

La bonne nouvelle c'est qu'Intel utilisera pour la partie Windows, le moteur d'exécution de Microsoft : **le Concurrency Runtime**.
Cette bibliothèque adopte le style standard de la Template Librairie (STL) tout en tirant parti des nouvelles caractéristiques du standard C++0x avec notamment les expressions lambda.
Voici un aperçu express des différentes fonctionnalités dont on peut distinguer quatre parties :

**a) Le parallélisme de tache**
o task_handle
o task_group

**b) Les primitives de synchronisation**
o reader_writer
o enter_critical
o Event

**c) Les algorithmes**
o parallel_for
o parallel_for_each
o parallel_invoke
o parallel_accumulate

**d) Les containeurs avec gestion de la concurrence**
o concurrent_queue
o concurrent_vector
o concurrent_unordered_map
o combinable

Il est à noter que le débogueur de Visual Studio a été remanié pour tenir compte du parallélisme avec une vision par thread par tâche etc.

Voilà pour l'essentiel.
je vous invite à consulter MSDN pour plus d'informations.

## 9. Conclusion

Beaucoup de nouveautés et de points positifs pour ce produit d'ores et déjà disponible.
Nouvelles fonctionnalités, confort d'utilisation de l'IDE amélioré, Visual Studio 2010 est en nette évolution par rapport à Visual Studio 2008.

Ce sera tout pour ce tour d'horizon rapide des nouveautés apportées à l'interface de la version 2010 de Visual Studio et à son module C++.

*Retrouvez l'article complet de Patrick OTTAVI en ligne : Lien104*

## Naissance du projet QExtend

Bonjour,

Nous sommes heureux de vous annoncer la création du projet QExtend : Lien105.

QExtend est une bibliothèque C++ développée par l'équipe Qt de la communauté Developpez.com. Son objectif est d'étendre et de simplifier l'utilisation de Qt ainsi que celle d'autres bibliothèques (Qwt, OpenCV...).

Le projet venant de commencer, il n'y a pas encore énormément de choses disponibles sur le repository. Toutefois, nous pouvons vous faire part d'une bonne liste de fonctionnalités prévues :
* pointeurs intelligents ;
* manipulateurs de layouts, signaux/slots et XML ;
* détection d'appui sur des touches du clavier ;
* classes d'aide pour QGraphics ;
* logger ;
* widgets, comme un menu de fichiers récemment ouverts ;
* CUDA ;
* Qwt ;
* trouver la grande question sur la vie, l'univers et le reste.

Et ce n'est qu'un petit résumé !

Si vous avez des questions, des remarques, des propositions, n'hésitez pas à nous les proposer sur ce thread. Si vous utilisez d'ores et déjà QExtend, nous vous proposons d'ajouter un tag [QExtend] au titre de votre message.

De plus, une fois connecté sur le gestionnaire de projets, vous pourrez aussi nous remonter les bogues rencontrés ou proposer des améliorations.

Si vous souhaitez participer au projet, manifestez-vous ici, nous vous recontacterons.

*Commentez cette news en ligne : Lien106*

## Sortie de Qt Mobility 1.0.0

Après un peu d'attente, voilà enfin la première sortie officielle de Qt Mobility.

Cette release contient 10 API dont 9 sont considérées comme finales et 1 comme bêta.
Les 9 API finales sont les suivantes :

- Service Framework
- Bearer Management
- Messaging
- Contacts
- Versit
- Publish and Subscribe
- Location
- System Information
- Sensors

L'API multimédia quant à elle, est à un très bon état de développement, mais l'équipe de développement se réserve la possibilité de modifier cette version. Dans ce cas, cette version bêta ne sera pas maintenue, il est donc à vos risques et périls de l'utiliser dès à présent.

**Où les trouver ?**

Elles sont disponibles sur la page Qt Solutions (Lien107) ainsi que dans la version bêta du SDK de Qt (Lien108).

Pour bien commencer, n'hésitez pas à parcourir le guide d'utilisation (Lien109).

**Et pour Symbian ?**

Cette release pour Maemo supporte uniquement Maemo PR 1.2 qui n'est pas encore paru.
Il y aura dans les semaines à venir un patch pour l'utilisation de Qt Mobility avec la plateforme Symbian.

**Voir aussi**

Le site Web de Qt (Lien110).

**Et vous ?**

Attendiez-vous avec autant d'impatience que nous cette sortie ? Prévoyez-vous d'utiliser ces API dans vos applications dans les mois à venir ?

*Commentez cette news en ligne : Lien111*

## Concevoir la prochaine génération d'interface utilisateur multiplateformes avec Qt

Le framework Graphics de Qt apporte différents outils permettant d'améliorer l'interface utilisateur de vos applications en modifiant l'apparence des objets graphiques et en les animant.
Cet article est une traduction autorisée de Building Next Generation UIs Across Platforms with Qt (Lien112), par Johan Thelin.

### 1. L'article original

Qt Quarterly est une revue trimestrielle électronique proposée par Qt à destination des développeurs et utilisateurs de Qt. Vous pouvez trouver les versions originales (Lien113).



Cet article est la traduction de l'article Building Next Generation UIs Across Platforms with Qt (Lien112) de Johan Thelin paru dans la Qt Quarterly Issue 32.

Cet article est une traduction d'un des tutoriels écrits par **Nokia Corporation and/or its subsidiary(-ies)** incluse dans la documentation de Qt, en anglais. Les éventuels problèmes résultant d'une mauvaise traduction ne sont pas imputables à Nokia.

### 2. Introduction

Avec Qt 4.6, une multitude de nouvelles classes ont été ajoutées. Toutes ces classes vont dans la même direction : des interfaces modernes, animées et réactives.

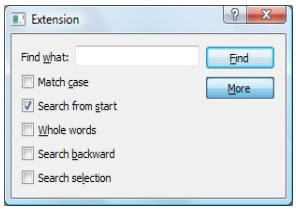
En regardant les interfaces utilisateur d'un point de vue utilisateur, les appareils mobiles actuels ont commencé à employer des interfaces utilisateur imitant le monde réel. Cela a permis d'ajouter des effets graphiques avancés tels que les transformations en trois dimensions, le flou, l'opacité, mais aussi les transitions fluides et la simulation physique. Dans ce type d'application, utiliser des blocs de construction de l'interface utilisateur standard n'est pas toujours désiré. Au contraire, l'interface utilisateur doit s'intégrer dans l'appareil physique qui a sa propre interface.

Qt permet d'aborder cette situation selon deux approches complémentaires. La première est constituée des méthodes d'interaction avec l'utilisateur. Qt supporte le multi-touch et la gestuelle en complément du clavier et de la souris. L'autre approche correspond aux graphiques, aux transitions et aux effets. Nous allons nous concentrer sur cette partie dans cet article.

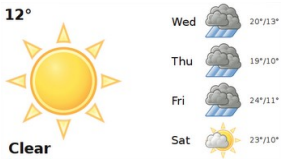
Les nouvelles classes de graphiques et d'animations de Qt 4.6 peuvent être divisées en trois groupes : les effets graphiques, les animations de propriétés et le framework de machine à états. Ces trois composants forment ensemble la base d'une interface utilisateur moderne et animée. Avant de pouvoir utiliser ces outils, nous avons besoin de donner à l'application une apparence correcte.

### 3. L'apparence de l'application

Traditionnellement, les applications de bureau sont fabriquées à partir de widgets rectangulaires. Les widgets gèrent leur propre contenu ainsi que le contenu de leurs enfants dans un rectangle qui leur est alloué. Ils sont également conçus pour être positionnés les uns par rapport aux autres et pour rester à peu près à la même place la plupart du temps. Le résultat est une interface plus ou moins rigide construite sur des rectangles. Les environnements de bureau jouent parfois avec les coins arrondis des boutons et ajoutent un peu de couleur. Aujourd'hui, les utilisateurs veulent plus.




*Classique*




*Moderne*

Pour contourner les limitations des widgets, il est possible d'utiliser le framework Graphics View (Lien114) comme base de votre interface utilisateur. Ce framework a remplacé l'ancienne API mais l'a aussi étendu. Avec Graphics View, il est possible de faire pivoter, de modifier l'échelle, de détacher et de déplacer des éléments, mais également d'appliquer des pseudo-transformations tridimensionnelles comme la rotation d'un objet autour de l'axe x ou y.

Au cours de l'évolution de Qt, de plus en plus de fonctionnalités ont été ajoutées, y compris la capacité à intégrer des widgets dans des scènes graphiques. Parmi les apports de Qt 4.6, on trouve un framework de classes pour l'animation, des effets graphiques, les états et des transitions pour l'interface utilisateur. En étant capable de mélanger des éléments graphiques avec des widgets et des animations, le résultat permet de créer des interfaces utilisateur modernes.

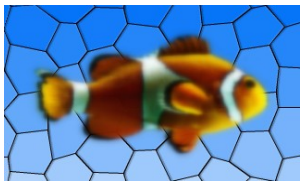
Lors de la création d'une interface utilisateur moderne, le travail des designers est d'obtenir la bonne combinaison d'effets. Le développeur qui implémente l'interface doit décider quel outil utiliser pour chaque partie du design. L'outil tout prêt est le framework Animation (Lien115). Ces classes définissent, regroupent et ordonnent les animations. En plus des animations, un ensemble d'effets, tels que le flou et l'ombrage, sont également disponibles.

Pour conduire une telle expérience de l'utilisateur, une sorte de machine à états est nécessaire. Cela est permis par le framework de machine à états de Qt (Lien116). Ici, plusieurs états peuvent être définis ainsi que des transitions entre eux avec des animations et des effets. Il n'est pas nécessaire de créer une liste de booléens et une variable globale représentant l'état courant associée à des minuteries et des slots utilisateurs. Une configuration unique d'une machine à états encapsule l'interface utilisateur complète avec des transitions et des animations couplées. Avec ce problème administratif réglé, les développeurs peuvent se concentrer sur l'ajout de fonctionnalités modernes à l'application.

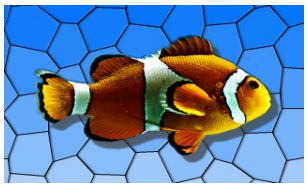
### 3.1. Les effets graphiques

L'une des caractéristiques nouvelles de Qt 4.6 est l'ensemble des nouvelles classes d'effets graphiques. La classe de base, QgraphicsEffect (Lien117), forme la base d'un framework ajoutant des effets tels que le flou et les ombres projetées. Ces effets peuvent être appliqués aux éléments d'interface utilisateur comme les widgets et les éléments graphiques. Il est possible de créer des effets personnalisés qui peuvent être utilisés dans les mêmes circonstances, ce qui signifie que vous pouvez faire ce que vous voulez.

Tous les effets graphiques sont contrôlés par un certain nombre de propriétés. Cela signifie non seulement que les effets peuvent être adaptés pour chaque application, mais aussi qu'ils peuvent être modifiés par le système d'animation que nous examinerons plus tard.



*Flou*

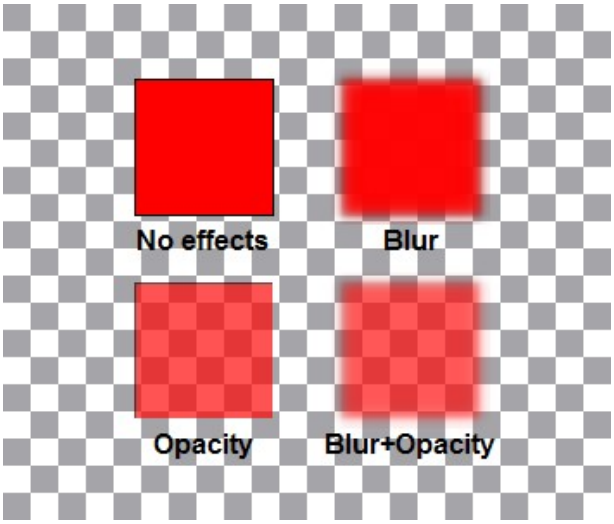


*Ombre projetée*

L'un des effets disponibles est l'effet de flou par la classe QgraphicsBlurEffect (Lien118). Elle applique un flou sur son élément source en fonction de la propriété blurRadius (Lien119). Un petit rayon donne peu de flou tandis qu'un grand rayon conduit à un flou plus important. Pour appliquer l'effet graphique à un élément graphique, utilisez la méthode QGraphicsItem::setGraphicsEffect() (Lien120) comme indiqué ci-dessous.

```
QGraphicsBlurEffect *effect = new
QGraphicsBlurEffect(this);
effect->setBlurRadius(0);

QGraphicsPixmapItem *item = new
QGraphicsPixmapItem();
// ...
item->setGraphicsEffect(effect);
```

Pour obtenir plus d'effets, il est possible de les combiner. Ceci est obtenu en ajoutant une hiérarchie aux objets graphiques et en appliquant un effet sur chaque niveau. Par exemple, des effets de flou et d'opacité sont appliqués ensemble dans l'exemple suivant. La classe QgraphicsRectItem (Lien121) est l'élément sur lequel nous voulons appliquer les effets tandis que la classe QgraphicsItemGroup (Lien121) est utilisé pour le second effet.

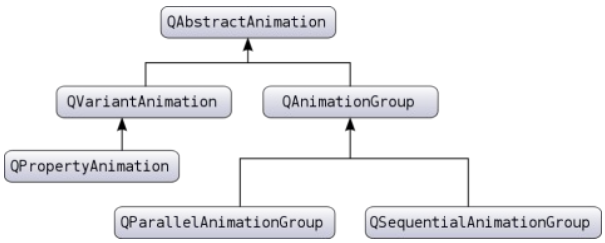



*Des rectangles avec des effets de flou et d'opacité*

### 3.2. Les classes d'animations

Un des éléments clés d'une interface utilisateur moderne est d'avoir des transitions animées et fluides au lieu des changements immédiats et soudains. Cela peut aider l'utilisateur à comprendre comment les éléments sont reliés entre eux et rendre l'interface utilisateur plus réaliste. Dernier point mais non des moindres, il contribue également à améliorer l'attrait visuel de l'ensemble du système.

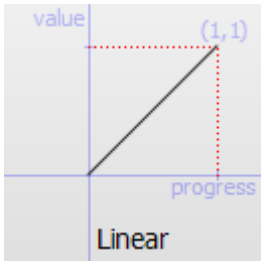
Qt 4.6 est livré avec un tout nouveau framework d'animation (Lien122). L'animation aurait pu être implémentée en utilisant des versions antérieures de Qt mais celle-ci aurait été limitée à la fourniture de time lines pour les animations et les classes animant des transformations d'objets graphiques. Le nouveau framework rend possible d'animer n'importe quelle propriété d'un Qobject (Lien123).

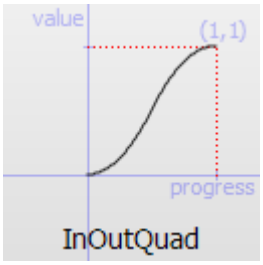


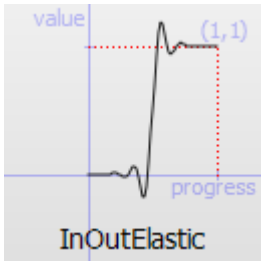

Le framework d'animation se compose de nombreuses classes basées sur la classe QabstractAnimation (Lien124). Fondamentalement, les classes utilisaient soit des outils facilitant la synchronisations des animations, soit des animations de propriétés. La synchronisation permet aux animations d'être exécutées en série ou en parallèle. La classe QpropertyAnimation (Lien125) est probablement celle que vous utiliserez le plus souvent. Elle permet d'animer individuellement les propriétés des Qobjects (Lien126).

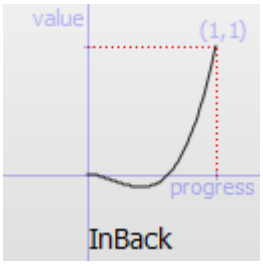
Lors de l'animation d'une propriété, les valeurs de début et de fin doivent être précisées. Cela est obtenu en spécifiant les QpropertyAnimation::setStartValue() (Lien127) et QpropertyAnimation::setEndValue() (Lien128) pour cette

propriété. Il est également possible de fixer des valeurs intermédiaires en utilisant la fonction QpropertyAnimation::setKeyValueAt() (Lien129). Lorsque vous définissez la valeur de propriété, l'animation est supposée fonctionner entre 0 et 1. Ce n'est pas tout à fait vrai puisque les animations peuvent utiliser les différentes courbes de relaxation. Ces courbes peuvent se déplacer légèrement en dehors de la gamme de 0 à 1 comme illustré ci-dessous.




Pour illustrer toutes ces techniques, le code ci-dessous montre une animation simple qui permet à un QpushButton (Lien130) d'apparaître dans un widget conteneur. Comme cette animation ne sera utilisée qu'une seule fois, l'argument QAbstractAnimation::DeleteWhenStopped (Lien131) est passé à la méthode de démarrage. Cela garantit que l'objet animation soit supprimé quand il est arrêté.

```
QPropertyAnimation *animation = new
QPropertyAnimation(button, "geometry");
animation->setStartValue(startRect);
animation->setEndValue(endRect);
animation-
>setEasingCurve(QEasingCurve::OutBounce);
animation-
>start(QAbstractAnimation::DeleteWhenStopped);
```

La même technique, montrée ici pour des widgets, peut être utilisée pour n'importe quel Qobject (Lien123). Cependant, la classe couramment utilisée pour cela, QabstactGraphicsItem (Lien132), n'hérite pas de Qobject (Lien123). Pour résoudre ce problème, un élément graphique personnalisé doit être créé dans ce cas. Ce n'est pas aussi lourd qu'il y paraît. Tout ce que la nouvelle classe doit faire est d'hériter de Qobject (Lien123) et d'une des classes dérivant de QgraphicsItem (Lien133) puis de rendre accessible la propriété voulue. Par exemple, la classe CustomRectItem ci-dessous rend accessible la propriété de rotation d'un QgraphicsRectItem (Lien134) pour le système d'animation. Aucun code supplémentaire n'est nécessaire, à l'exception d'un constructeur court et de la ligne Q_PROPERTY (Lien135).

```
class CustomRectItem : public QObject, public
QGraphicsRectItem
{
    Q_OBJECT
    Q_PROPERTY(qreal rotation READ rotation WRITE
setRotation)

public:
    CustomRectItem(const QRectF &rect)
        : QObject(0),QGraphicsRectItem(rect)
    { }
};
```

Compte tenu de la définition de la classe CustomRectItem, la rotation de l'objet est aussi simple que de créer un objet QpropertyAnimation (Lien136) pour cette classe et de le démarrer.

```
QPropertyAnimation *animation = new
QPropertyAnimation(item, "rotation", this);
animation->setStartValue(0);
animation->setEndValue(90);
animation->start();
```

### 3.3. Ajouter des états

Un cas habituel d'utilisation des animations est de changer entre différents états de l'interface utilisateur comme une vue en liste, une vue détaillée et un mode d'édition. La plupart des développeurs ont entendu parler, et même peut-être déjà utilisé, des machines à états. Celles-ci sont parfaites pour définir des états et les transitions entre ces états.

Qt 4.6 est livré avec un tout nouveau framework de machine à états (Lien137) qui vous permet de faire cela. Grâce à ce framework, la valeur des propriétés des objets peut être définie pour un ensemble d'états. Une transition peut ensuite être ajoutée entre les différents états et l'ensemble peut alors être démarré.

Un exemple simple montrant cela peut être construit à partir de nos exemples précédents. Nous prenons une scène graphique avec deux rectangles inclus dans des groupes (pour utiliser deux effets). Nous définissons ensuite deux états, le premier dans lequel le rectangle vert apparaît opaque à 100% sans flou tandis que le rectangle jaune est opaque à 50% et flou, le second état où ces effets sont appliqués au rectangle vert au lieu du jaune.

La première chose pour créer cela est de mettre en place la scène : vous pouvez voir comment faire dans le code source disponible en téléchargement à la fin de cet article. Les résultats produits sont des rectangles et des groupes appelés greenRect, yellowRect, greenGroup et yellowGroup. Nous pouvons ensuite leur appliquer des effets puis ajouter ces éléments à la scène.

```
QGraphicsOpacityEffect *greenOpacity = new
QGraphicsOpacityEffect();
QGraphicsOpacityEffect *yellowOpacity = new
QGraphicsOpacityEffect();
```

```
QGraphicsBlurEffect *greenBlur = new
QGraphicsBlurEffect();
QGraphicsBlurEffect *yellowBlur = new
QGraphicsBlurEffect();

greenRect->setGraphicsEffect(greenOpacity);
yellowRect->setGraphicsEffect(yellowOpacity);
greenGroup->setGraphicsEffect(greenBlur);
yellowGroup->setGraphicsEffect(yellowBlur);
```

Quand tout cela est en place, le framework de machine à états entre en scène. Nous créons deux états, nommés greenState et yellowState. L'initialisation de l'état greenState est indiquée ci-dessous. Notez que les propriétés modifiées par cet état sont définies à l'aide de la méthode assignProperty() (Lien138) qui prend un pointeur d'objet, un nom de propriété et une valeur. La valeur est traitée comme un Qvariant (Lien139), donc vous pouvez utiliser ce que vous voulez dedans. Vous pouvez même ajouter le support de vos propres types en utilisant le système de type de Qt, mais je m'égare.

```
QState *greenState = new QState();
greenState->assignProperty(greenOpacity,
"opacity", 1);
greenState->assignProperty(greenBlur,
"blurRadius", 0);
greenState->assignProperty(yellowOpacity,
"opacity", 0.5);
greenState->assignProperty(yellowBlur,
"blurRadius", 5);
```

Pour chaque état, une série de transitions doit être ajoutée. Ces transitions permettent de passer de l'état actuel vers un autre état. Comme nous n'avons que deux états, ça nous laisse avec une transition par état.

Avant de regarder le code, nous allons discuter des transitions. Toutes les transitions héritent de la classe QabstractTransition (Lien140). À ce niveau, les états d'origine et de fin peuvent être spécifiés. Il y a donc des spécialisations de cette classe abstraite représentant une transition en fonction de la façon dont la transition est déclenchée. Par exemple, il y a des transitions déclenchées par un événement (QeventTransition (Lien141)) qui peut intercepter différents événements : l'appui sur une touche spécifique du clavier (QkeyEventTransition (Lien142)) ou une action de la souris (QmouseEventTransition (Lien143)). Il y a également la classe QsignalTransition (Lien144) qui implémente une transition déclenchée par un signal spécifique. C'est la classe que nous utilisons dans le code ci-dessous.

```
QSignalTransition *t = new
QSignalTransition(this, SIGNAL(yellowClicked()));
t->setTargetState(yellowState);
greenState->addTransition(t);
```

Maintenant que tous les états sont créés, ils doivent être ajoutés à l'objet QstateMachine (Lien145) qui gère l'ensemble. Comme vous pouvez le voir, nous ajoutons tous les états dans la machine à états mais un seul comme étant l'état initial. Cela est nécessaire pour donner à la machine à états un point de départ (n'oubliez pas que toutes les transitions ont un état d'origine).

```
QStateMachine *machine = new QStateMachine(this);
machine->addState(greenState);
machine->addState(yellowState);
machine->setInitialState(greenState);
```

Maintenant, il est temps de dire à la machine à états comment changer d'un état à un autre. C'est là que les classes d'animation que vous avez vues plus tôt entrent en jeu. Il est possible de préciser exactement quelle animation utiliser pour chaque objet, propriété et transition. Mais pour faire avancer les choses, vous pouvez également spécifier une animation par défaut pour chaque objet et propriété à un niveau de la machine à états.

```
machine->addDefaultAnimation(new
QPropertyAnimation(greenBlur, "blurRadius"));
machine->addDefaultAnimation(new
QPropertyAnimation(yellowBlur, "blurRadius"));
machine->addDefaultAnimation(new
QPropertyAnimation(greenOpacity, "opacity"));
machine->addDefaultAnimation(new
QPropertyAnimation(yellowOpacity, "opacity"));
```

Avant de pouvoir démarrer l'application et profiter des transitions amusantes, il reste une étape finale : démarrer la machine à états.

```
machine->start();
```

## 4. Qu'en est-il des widgets ?

Jusqu'à présent, nous nous sommes concentrés sur les vues graphiques et les éléments, mais presque tout le monde utilise une interface utilisateur basée sur les widgets. La perspective de devoir remplacer tout le travail déjà accompli semble être une tâche assez lourde. La bonne nouvelle est que presque tous les effets peuvent aussi bien être utilisés dans un environnement basé sur les widgets.

Commençons par un exemple. Regardons les deux états de dialogue ci-dessous. Ici, les widgets inutilisés sont placés en dehors du widget contenant. Une autre option aurait été de modifier la visibilité des éléments.


Ces états sont traduits dans le code ci-dessous par le biais des variables viewState et editState. Les états spécifiques à chaque état sont définis par des appels à assignProperty() (Lien146). Comme vous pouvez le voir, pour rendre possible l'utilisation de ces effets, vous devez sacrifier les layouts. C'est malheureux, mais en explorant les différentes possibilités, vous trouverez le bon mélange entre le bricolage et la flexibilité des layouts. (NDLT : il est maintenant possible d'utiliser des layouts avec la machines à états en masquant ou affichant les objets à l'aide de la propriété visible des Qwidget (Lien147)).

```
QState *viewState = new QState();
viewState->assignProperty(editButton, "geometry",
QRect(180, 5, 70, 30));
```

```
viewState->assignProperty(okButton, "geometry",
QRect(110, -35, 70, 30));
viewState->assignProperty(cancelButton,
"geometry", QRect(180, -35, 70, 30));
viewState->assignProperty(m_textLabel,
"geometry", QRect(5, 5, 175, 30));
viewState->assignProperty(m_textEdit, "geometry",
QRect(-110,5, 105, 30));

QState *editState = new QState();
editState->assignProperty(editButton, "geometry",
QRect(180, 55, 70, 30));
editState->assignProperty(okButton, "geometry",
QRect(110, 5, 70, 30));
editState->assignProperty(cancelButton,
"geometry", QRect(180, 5, 70, 30));
editState->assignProperty(m_textLabel,
"geometry", QRect(-200, 5, 175, 30));
editState->assignProperty(m_textEdit, "geometry",
QRect(5,5, 105, 30));
```

Maintenant que les états ont été définis, il est temps d'ajouter les transitions.

```
viewState->addTransition(editButton,
SIGNAL(clicked()), editState);
editState->addTransition(okButton,
SIGNAL(clicked()), viewState);
editState->addTransition(cancelButton,
SIGNAL(clicked()), viewState);
```

Nous ajoutons ensuite les états au gestionnaire avant d'ajouter les animations par défaut. Les widgets m_textLabel et m_textEdit sont déplacés à l'aide de la courbe standard tandis que les boutons se déplacent selon un chemin plus "dynamique" en utilisant la courbe QeasingCurve::InOutBack (Lien148).

```
QStateMachine *machine = new QStateMachine(this);
machine->addState(viewState);
machine->addState(editState);
machine->setInitialState(viewState);

QPropertyAnimation *pa;
machine->addDefaultAnimation(pa = new
QPropertyAnimation(editButton, "geometry"));
pa->setEasingCurve(QEasingCurve::InOutBack);

// ...

machine->addDefaultAnimation(new
QPropertyAnimation(m_textEdit, "geometry"));
```

Après avoir défini tous les états et les transitions, l'ensemble de l'interface utilisateur prend vie dès que l'instance de QstateMachine (Lien149) est démarrée. Vos utilisateurs devraient avoir une agréable surprise en cliquant sur le bouton Edit pour la première fois.

## 5. Divers

Le code source de l'exemple présenté dans cet article est disponible au téléchargement à l'adresse suivante : qq32-blurstates.zip (Lien150).

Johan Thelin est l'auteur du livre *Foundations of Qt Development* de chez Apress et a un faible pour les systèmes embarqués. Il est également impliqué sur le site QtCentre ainsi que dans le développement de logiciels et la rédaction technique.

*Retrouvez l'article de Johan Thelin traduit par Guillaume Belz en ligne : Lien151*


## Mac OS X Snow Leopard Précis et Concis

Snow Leopard, la nouvelle version du désormais célèbre Mac OS X n'abonde pas, cette fois, en nouveautés. Cette nouvelle version est une amélioration notable du sytème Leopard, tant du point de vue des technologies employées (comme Grand Central Dispatch) que des applications qui ont été revisitées et offrent une plus grande souplesse d'utilisation au quotidien.

Le système s'est fait plus léger et rapide. Le Mac prend en charge Microsoft Exchange, le Finder a été largement amélioré, les applications comme Aperçu, iChat, QuickTime, Automator ont également été revisitées. Le menu Services propose plus de fonctionnalités, etc.

Cette sixième édition est une visite guidée du système. Ouvrage de référence conçu dans le but de vous aider à maîtriser et optimiser votre Mac, vous y apprendrez d'abord à installer votre système et migrer vos données.

Le troisième chapitre est une prise en main : vous y découvrirez le Finder, le compte utilisateur, les diverses fonctionnalités de sécurité.

Le quatrième chapitre est consacré à l'interface : du Dock au Finder, aux Services ou à Exposé et Dashboard, tout est passé en revue.

Le chapitre 5 vous permettra d'appréhender les Préfèrences Système, à partir desquelles vous aurez le loisir de personnaliser votre Mac.

Les applications et utilitaires présents sur votre Mac sont ensuite détaillés un par un.

L'auteure termine sur un chapitre de questions/réponses qui fournit à la fois des astuces et des solutions en cas de soucis.

Ce Précis et concis est un condensé de ce qu'il faut savoir sur Snow Leopard. Il en dessine la cartographie à l'aide d'explications synthétiques et répond aux besoins immédiats des utilisateurs.

**Critique du livre par Marcos Ickx**

Contrairement à Mac OS X Leopard, Précis et concis où Nicoletis Nathalie avait traduit le livre écrit par Chuck Toporek, il s'agit ici d'une version originale et non d'une traduction du livre Mac OS X Snow Leopard Pocket Guide écrit cette fois-ci par Chris Seibold.

Les livres qui sont publiés aux éditions Digit Books ne le sont, comme le laisse supposer le nom de la maison d'édition, qu'au format numérique.

C'est donc au format PDF que j'ai reçu l'exemplaire de ce livre. PDF qui ne contient aucune protection de type DRM. Ce qui signifie que vous pouvez installer ce PDF où vous voulez. Que ce soit sur votre Mac, votre iPhone, votre iPad, votre PC, liseuse électronique.

Par contre, pour éviter tout de même que le PDF se retrouve un peu partout dans la nature, l'adresse email de la personne qui a acheté le livre apparaît au pied de chaque page.

Cela n'est aucunement dérangeant puisque la mise en page du livre tient compte du fait que l'adresse email sera rajoutée au bas de chaque page.

En 310 pages, Nicoletis Nathalie nous fait passer en revue ce nouveau système d'exploitation qu'est Mac OS X Snow Leopard.

Les sujets abordés sont vastes, les nouveautés de Mac OS X Snow Leopard bien traitées, et la mise en page rend ce livre agréable à lire à l'écran.

Le fait que ce livre soit au format PDF permet d'utiliser le moteur de recherche pour y retrouver facilement ce qu'on cherche. Et on profite de plus des copies d'écran en couleur, ce qui n'aurait certainement pas été le cas si le livre avait été disponible au format imprimé.

Je voulais également partager avec vous. Ô combien j'ai été agréablement surpris de la réactivité de Digitbooks ! En effet, une fois que j'ai eu fini la lecture du livre, j'ai envoyé à DigitBooks quelques petites remarques.

Eh bien, figurez-vous que peu de temps après je recevais une réponse de DigitBooks disant que certaines remarques avait été prises en compte et m'envoyait une nouvelle version du PDF où la majorité de mes remarques (et d'autres) avaient été prises en compte.

Et là, je dis bravo.

Si vous n'avez pas encore de livre consacré à Mac OS X Snow Leopard à votre disposition, n'hésitez pas une seconde à vous procurer ce livre qui vous permettra de découvrir les nouveautés apparues avec Snow Leopard (et elles sont plus nombreuses que ce que l'on pourrait croire).

Le prix de 11€ n'est vraiment pas exagéré. De plus, il faut savoir que pour ce prix vous avez non seulement le livre au format PDF, mais également au format MobiPocket, et au format epubs (format reconnu par la plupart des lecteurs de livres électroniques, par le Kindle, par l'iPad, l'iPhone...).

Pour 1€ de plus, vous pouvez également feuilleter le livre en ligne depuis la liseuse en ligne du site immateriel.

Et si vous ne voulez lire ce livre que depuis la liseuse en ligne du site immateriel, il ne vous en coutera que 5€, ce qui est vraiment un prix ridiculement bas.

**Critique du livre par kOrt3x**

Après avoir lu plusieurs livres sur Mac OS X, celui là est différent.

Rien à voir avec les livres qui expliquent pendant des chapitres comment paramétrer son client de messagerie ou configurer son Time Machine.

Il n'est pas fait pour expliquer la naissance de Mac OS X avec son histoire et l'historique de son entreprise qui l'a créé, mais vraiment pour expliquer tous les petits détails de Mac OS X.

Certes il explique les nouveautés et évolutions de Mac OS X, comme le support de Microsoft Exchange, l'adressage 64 bits, Grand Central Dispatch, etc.
Tous les détails sont passés au peigne fin, voire même des fois un peu trop.

Les préférences système sont très bien détaillées : elles sont décrites une par une, du fond d'écran au FireWall.
Elles y passent toutes.

Pratique pour savoir exactement où aller pour modifier un paramètre système.

Les raccourcis clavier principaux sont décrits, des plus simples aux plus complexes.

Les principaux éléments d'interface, comme le Dock, le Finder, Exposé ou la barre des menus sont très bien développés, voire un peu trop, comme le Dashboard qui est décrit dans les moindres coins, car tous les widgets installés par défaut sont développés.

Toutes les applications principales d'Apple et les utilitaires sont détaillés avec une simple description sans rentrer dans les détails de chaque application.

Vous trouverez une très bonne partie de questions/réponses avec une série de questions toutes simples pour certains utilisateurs de Mac OS X, mais qui peuvent être très pratiques pour les nouveaux utilisateurs de Mac.

Puis l'ouvrage se termine avec une partie "Ressources" qui référence une liste de logiciels pouvant compléter votre éventail d'applications ou le manque de certains logiciels pouvant être utilisés sous Windows ou Linux, comme des traitements de texte, des clients FTP et bien d'autres.

En résumé, cet ouvrage serait plus orienté vers les débutants ou les futurs utilisateurs de Mac OS X.

*Retrouvez cette critique de livre sur la page livres Mac : Lien152*

# Simulacres de tests avec EasyMock et JUnit 4

Cet article va vous présenter l'utilisation de Easymock et de JUnit 4 pour effectuer des tests unitaires avec des simulacres de tests.

## 1. Introduction

Les objets simulacres permettent d'effectuer des tests unitaires sur un objet dépendant d'autres objets. On va remplacer ces objets dont dépend l'objet à tester par des simulacres. On va, par exemple pouvoir vérifier que la méthode xyzzy() a été appelée 5 fois et a retourné 33. Cela peut être pratique dans bien des cas. Par exemple si l'objet réel (celui qu'on mock) est lent ou non-déterministe (dépendant du temps ou même de la météo). Ces objets sont très difficiles à tester car on pourrait faire plein de tests sans jamais tomber sur les cas spéciaux. Les cas de tests nous permettront de traiter ces cas spéciaux.

Il existe plusieurs outils permettant de faire des objets simulacres. Dans cet article, nous allons utiliser EasyMock 2.5.2 pour effectuer ces tests. Pour ce qui est des tests, nous allons utiliser JUnit 4.7.

Voici l'interface à tester :

```
public interface ISimpleDao {
        void save(String title);
        void remove(String title) throws
NotExistingException;
        int count();
        void debug();
        boolean isValid(String title);
        void insert(String title);
}
```

Et voici notre classe à tester :

```
public class SimpleService {
        private ISimpleDao dao;

        public void setDao(ISimpleDao dao){
                this.dao = dao;
        }

        public void insert(String title){
                if(dao.isValid(title)){
                        dao.insert(title);
                }
        }

        public void save(String... titles){
                for(String title : titles){
                        dao.save(title);
                }
        }

        public boolean remove(String title){
                try {
                        dao.remove(title);
                } catch (NotExistingException e){
                        return false;
                }

                return true;
        }

        public int size(){
                return dao.count();
        }

        public void debug(){
                System.out.println("Debug
information of SimpleService");
                dao.debug();
        }
}
```

Notre mock va donc implémenter l'interface ISimpleDao et nous allons le passer à SimpleService qui est la classe à tester dans notre cas. Cet exemple est vraiment simpliste. Dans les cas pratiques, vous ferez face à des cas beaucoup plus complexes, mais cet exemple permettra de couvrir la plupart des fonctionnalités d'EasyMock.

La vérification se fait essentiellement via deux méthodes :

- expect(T value) : permet de spécifier une valeur de retour attendue ;
- expectLastCall() : à utiliser pour les méthodes ne retournant rien.

Ces deux méthodes renvoient un objet IExpectationSetters qui permet de configurer ce que l'on attend de la méthode mockée, comme par exemple, le nombre de fois qu'elle doit être appelée.

## 2. Vérifier un comportement

Voici la structure de base pour notre classe de test :

```
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.*;
import static org.easymock.EasyMock.*;

public class SimpleServiceTest {
    private SimpleService simpleService;
    private ISimpleDao simpleDaoMock;

    @Before
    public void setUp(){
        simpleService = new SimpleService();
        simpleService.setDao(simpleDaoMock);
    }
```

```
    @Test
    public void insertValid(){}

    @Test
    public void insertNotValid(){}

    @Test
    public void save(){}

    @Test
    public void removeWithoutException() throws
NotExistingException {}

    @Test
    public void removeWithException() throws
NotExistingException {}

    @Test
    public void size(){}

    @Test
    public void debug(){}
}
```

Premièrement, nous allons commencer par créer un objet simulacre (un mock). Pour cela, il nous faut utiliser la classe EasyMock et sa méthode createMock() qui prend en paramètre l'interface que doit implémenter le mock. Pour améliorer la clarté du code, on va utiliser un import statique comme pour JUnit.

```
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.*;
import static org.easymock.EasyMock.*;

public class SimpleServiceTest {
    private SimpleService simpleService;
    private ISimpleDao simpleDaoMock;

    @Before
    public void setUp(){
        simpleDaoMock =
createMock(ISimpleDao.class);

        simpleService = new SimpleService();
        simpleService.setDao(simpleDaoMock);
    }
}
```

Cela va simplement créer un objet mock implémentant l'interface ISimpleDao. La première action que nous pouvons entreprendre avec EasyMock est de vérifier qu'une méthode a bien été appelée. Avec EasyMock, cela fonctionne comme un enregistrement :

- on joue la séquence désirée sur l'objet mock ;
- on enregistre la séquence jouée ;
- on teste l'objet ;
- on vérifie si la séquence a été correctement rejouée.

On va donc simplement tester pour commencer si la méthode debug() de SimpleService appelle bien la méthode debug() de notre classe DAO (Data Access Object) :

```
@Test
public void debug(){
    simpleDaoMock.debug();

    replay(simpleDaoMock);

    simpleService.debug();

    verify(simpleDaoMock);
}
```

La méthode replay() permet de sauvegarder l'enregistrement et la méthode verify() permet de vérifier que ce qui est fait après replay() est bien conforme à l'enregistrement. Si vous lancez le test, il va être validé. Maintenant, si on commente dao.debug() dans SimpleService, le test ne va pas se dérouler correctement :

```
java.lang.AssertionError:
  Expectation failure on verify:
    debug(): expected: 1, actual: 0
        at
org.easymock.internal.MocksControl.verify(MocksCo
ntrol.java:111)
        at
org.easymock.EasyMock.verify(EasyMock.java:1608)
        at
com.dvp.wichtounet.easymock.SimpleServiceTest.deb
ug(SimpleServiceTest.java:45)
        at
sun.reflect.NativeMethodAccessorImpl.invoke0(Nati
ve Method)
        at
sun.reflect.NativeMethodAccessorImpl.invoke(Nativ
eMethodAccessorImpl.java:57)
        at
sun.reflect.DelegatingMethodAccessorImpl.invoke(D
elegatingMethodAccessorImpl.java:43)
        at
org.junit.runners.model.FrameworkMethod$1.runRefl
ectiveCall(FrameworkMethod.java:44)
        at
org.junit.internal.runners.model.ReflectiveCallab
le.run(ReflectiveCallable.java:15)
        at
org.junit.runners.model.FrameworkMethod.invokeExp
losively(FrameworkMethod.java:41)
        at
org.junit.internal.runners.statements.InvokeMetho
d.evaluate(InvokeMethod.java:20)
        at
org.junit.internal.runners.statements.RunBefores.
evaluate(RunBefores.java:28)
        at
org.junit.runners.BlockJUnit4ClassRunner.runChild
(BlockJUnit4ClassRunner.java:76)
        at
org.junit.runners.BlockJUnit4ClassRunner.runChild
(BlockJUnit4ClassRunner.java:50)
        at
org.junit.runners.ParentRunner$3.run(ParentRunner
.java:193)
        at
org.junit.runners.ParentRunner$1.schedule(ParentR
unner.java:52)
        at
org.junit.runners.ParentRunner.runChildren(Parent
Runner.java:191)
        at
org.junit.runners.ParentRunner.access$000(ParentR
unner.java:42)
        at
```

```
org.junit.runners.ParentRunner$2.evaluate(ParentR
unner.java:184)
        at
org.junit.runners.ParentRunner.run(ParentRunner.j
ava:236)
        at
org.junit.runner.JUnitCore.run(JUnitCore.java:157
)
        at
com.intellij.junit4.JUnit4IdeaTestRunner.startRun
nerWithArgs(JUnit4IdeaTestRunner.java:94)
        at
com.intellij.rt.execution.junit.JUnitStarter.prep
areStreamsAndStart(JUnitStarter.java:165)
        at
com.intellij.rt.execution.junit.JUnitStarter.main
(JUnitStarter.java:60)
        at
sun.reflect.NativeMethodAccessorImpl.invoke0(Nati
ve Method)
        at
sun.reflect.NativeMethodAccessorImpl.invoke(Nativ
eMethodAccessorImpl.java:57)
        at
com.intellij.rt.execution.application.AppMain.mai
n(AppMain.java:110)
```

EasyMock détecte donc bien que notre méthode n'a pas été appelée au contraire de ce qui a été spécifié par l'enregistrement et, de ce fait, le test échoue. Si vous essayez de vérifier si des méthodes non-void ont bien été appelées de la même manière, vous devriez avoir une exception de type IllegalStateException. En effet, pour les méthodes retournant quelque chose, il faut indiquer à EasyMock ce qu'il faut retourner. Nous allons voir cela au prochain chapitre.

### 3. Attendre des valeurs de retour

Nous allons maintenant traiter des méthodes qui retournent quelque chose. Dans ce cas, il faut définir un comportement pour pouvoir vérifier ce comportement. Pour ce faire, il faut utiliser la méthode expect() et andReturn() pour spécifier une valeur de retour. Voici comment cela pourrait s'écrire pour le test de la méthode size() :

```
@Test
public void size(){
    expect(simpleDaoMock.count()).andReturn(32);

    replay(simpleDaoMock);

    assertEquals(32, simpleService.size());

    verify(simpleDaoMock);
}
```

Par ces quelques lignes de code, on fait deux tests. On vérifie si la méthode count() a bien été appelée et si size() retourne la même valeur que count(). Ce qui est bien le cas si on lance le test. On vient donc voir qu'il est très simple de spécifier une valeur de retour pour une méthode sur un objet mock.

### 4. Traiter les exceptions

EasyMock permet également de traiter les exceptions. Il nous faudra à nouveau utiliser la méthode expect(), mais cette fois, au lieu de spécifier une valeur de retour, on va spécifier une exception qu'il faut lever avec la méthode andThrow(). Voyons ce que ça donnerait avec le test de la méthode remove() avec et sans exception :

```
//Sans exception
@Test
public void removeWithoutException() throws
NotExistingException {
    simpleDaoMock.remove("Mary");

    replay(simpleDaoMock);

    assertTrue(simpleService.remove("Mary"));

    verify(simpleDaoMock);
}

//Avec exception
@Test
public void removeWithException() throws
NotExistingException {
    simpleDaoMock.remove("Arthur");

    expectLastCall().andThrow(new
NotExistingException());

    replay(simpleDaoMock);

    assertFalse(simpleService.remove("Arthur"));

    verify(simpleDaoMock);
}
```

Encore une fois, il nous a suffi d'un seul appel de méthode pour spécifier une exception et, ensuite, nous avons pu vérifier le comportement de notre service en fonction du comportement de notre mock.

### 5. Divers

#### 5.1. Vérifier le nombre d'appels

Testons maintenant notre méthode save() :

```
@Test
public void save(){
    simpleDaoMock.save("xyzzy");
    simpleDaoMock.save("xyzzy");
    simpleDaoMock.save("xyzzy");
    simpleDaoMock.save("xyzzy");
    simpleDaoMock.save("xyzzy");

    replay(simpleDaoMock);

    simpleService.save("xyzzy", "xyzzy", "xyzzy",
"xyzzy", "xyzzy");

    verify(simpleDaoMock);
}
```

Ce genre de code devient vite très lourd à écrire en fonction du nombre d'appels. On a deux solutions : soit on fait une boucle pour appeler les méthodes du mock, soit on utilise les fonctionnalités d'EasyMock qui permettent de spécifier le nombre de fois qu'une méthode doit être appelée grâce à la méthode times() :

```
@Test
public void save(){
    simpleDaoMock.save("xyzzy");

    expectLastCall().times(5);

    replay(simpleDaoMock);

    simpleService.save("xyzzy", "xyzzy", "xyzzy",
"xyzzy", "xyzzy");

    verify(simpleDaoMock);
}
```

Beaucoup plus clair, non ? Il est également possible de spécifier qu'une méthode peut être appelée un nombre indéfini de fois avec la méthode anyTimes() ou encore un certain nombre de fois compris dans un intervalle avec la méthode times(min, max).

### 5.2. Vérifier l'ordre des appels

On va maintenant tester notre méthode insert() :

```
@Test
public void insertValid(){
    expect(simpleDaoMock.isValid("Arthur")).andRe
turn(true);

    simpleDaoMock.insert("Arthur");

    replay(simpleDaoMock);

    simpleService.insert("Arthur");

    verify(simpleDaoMock);
}

@Test
public void insertNotValid(){
    expect(simpleDaoMock.isValid("Arthur")).andRe
turn(false);

    replay(simpleDaoMock);

    simpleService.insert("Arthur");

    verify(simpleDaoMock);
}
```

Vous allez me dire qu'on a déjà vu tout cela, certes, mais dans ce genre de cas, il peut également être intéressant de vérifier que les appels se font au bon endroit. En effet, si la méthode isValid() est appelée après que l'insert() a été fait, elle n'a pas beaucoup d'intérêt. Avec EasyMock, il y a deux façons de vérifier l'ordre des appels. Soit on utilise la méthode createStrictMock() au lieu de createMock() ou alors on utilise checkOrder(mock, true) pour activer la vérification de l'ordre. Un mock strict est un bouchon qui va vérifier l'ordre des appels. On n'a donc pas besoin de changer nos tests, il suffit simplement d'utiliser une de ces deux méthodes dans notre méthode before().

### 5.3.Mocker une classe

EasyMock met à disposition une extension pour créer un mock d'une classe et non d'une interface. Il s'agit d'EasyMock Class Extension. L'utilisation de base reste la même, il faut juste changer l'import :

```
import static
org.easymock.classextension.EasyMock.*;
```

En plus de cela, il est également possible d'effectuer un mocking partiel en ne mockant par exemple qu'une seule méthode :

```
Mocked mock =
createMockBuilder(Mocked.class).addMockedMethod("
mockedMethod").createMock();
```

Il faut toutefois faire attention au fait que les classes finales ne sont pas supportées. Si la classe contient des méthodes finales, elles ne seront pas mockées et elles seront appelées normalement.

## 6. Conclusion

Voilà, nous avons maintenant passé en revue les principales fonctionnalités que nous offre EasyMock pour la création d'objets mocks pour les tests unitaires. Comme vous avez pu le constater, c'est un moyen simple mais très puissant de vérifier le comportement d'un objet en fonction du comportement d'un objet dont il dépend. Il existe d'autres librairies qu'EasyMock pour faire cela comme JMock, JMockit ou encore Mockito. Personnellement, je trouve qu'EasyMock est la plus agréable à utiliser et fournit toutes les fonctions dont j'ai besoin pour faire mes tests, c'est pourquoi j'ai choisi de présenter cette librairie.

Vous pouvez télécharger les sources de cet article ici : Lien153

Si vous êtes intéressé par un autre framework, je vous invite à lire cet article sur Jmockit (Lien154), de Florence Chabanois.

Pour vos questions sur les outils de test Java, vous pouvez consulter le forum dédié (Lien155).

Pour plus d'informations sur EasyMock, je vous invite à consulter la documentation officielle en Français (Lien156).

*Retrouvez l'article de Baptiste Wicht en ligne : Lien157*

## Améliorer la testabilité

Cet article donne quelques pistes de réflexion pour faciliter les tests dans le cadre d'une méthode de développement "cycle en V".

### 1. Introduction

#### 1.1. De quoi s'agit-il ?

Un produit peut être plus ou moins facile à tester. Qui n'a pas été confronté un jour à ce type de problèmes :

1. Un logiciel sans spécifications ou des spécifications peu fiables : comment écrire le plan de test dans ces conditions ?
2. Une anomalie est détectée mais il est impossible de localiser le problème.
3. Une architecture peu adaptée aux tests unitaires.

Cet article a pour objectif d'étudier quels sont les points qui vont faciliter notre travail de testeur. Plusieurs éléments vont contribuer à améliorer la testabilité à différentes phases du processus de test d'un cycle en V.

#### 1.2. Quelques définitions

1. Tests unitaires : tests de composants logiciels individuels faits par les développeurs.
2. Tests d'intégration : tests permettant de valider les interfaces entre composants. Ces tests sont généralement effectués par l'équipe de développement ou par une équipe dédiée dans le cas d'une intégration complexe.
3. Tests système : tests permettant de valider un système intégré par rapport à des exigences spécifiques. Ces tests sont généralement effectués par une équipe de validation dédiée ou par l'équipe de développement. Ces tests couvrent les aspects fonctionnels et non fonctionnels (performance, installation, robustesse, sauvegarde ...)
4. Recette : processus qui permet au client de valider que la livraison des fonctionnalités développées par le fournisseur, correspond au cahier des charges.

### 2. Stratégie de test

Deux éléments de la stratégie de test concernent la testabilité :

1. Quels sont les objectifs qualité de nos tests ?
2. De quels moyens allons-nous avoir besoin ?

#### 2.1. Quoi ?

Quand on parle de tester un produit le premier élément que nous exigeons est les spécifications fonctionnelles et pourtant il me semble encore plus important dans un premier temps de bien cerner quels sont les objectifs de nos futures campagnes de tests. Ces éléments sont essentiels pour calibrer nos efforts de test dans la bonne direction.
Imaginons que nous devons tester un nouveau service rendu par une interface Web. Nous ne sommes pas les premiers sur ce marché et la concurrence est rude. Il est d'autre part vital pour la société d'obtenir rapidement 5 % de part de marché. Les éléments stratégiques concernant le produit sont donc connus. Ils sont essentiels pour porter les efforts de test au bon endroit.
L'ergonomie doit permettre au client de trouver facilement les services qu'il désire, et les accès au site doivent être rapides sous peine de voir le client se tourner vers un site équivalent.
En plus des tests fonctionnels classiques, il va falloir valider la performance des interfaces et leur ergonomie. Ces tests demandent des compétences particulières qu'il faudra prévoir. À ce stade nous voyons également qu'il faudra un environnement de test et des moyens dédiés. Ces éléments qui seront traités dans le chapitre suivant, contribuent également à la testabilité du produit.

#### 2.2. Comment ?

Nous voulons mettre notre logiciel en situation réelle et simuler tous les scénarios d'utilisation et d'erreur possibles. En règle générale, nous ne travaillons pas dans le monde réel mais dans un environnement simulé qui se rapproche du réel. Nos outils devront :

1. Être capables de simuler un environnement réel en mode nominal, erreur et charge.
2. Être fiables c'est-à-dire eux-mêmes testés.

Simuler un environnement externe peut être complexe, ces outils peuvent se trouver sur le marché comme par exemple un simulateur de réseau GSM pour tester un mobile. Dans ce cas, l'outil est connu et testé. Il suffit de faire une étude pour choisir celui qui conviendra le mieux (fonctionnalités offertes, type de tests, profil utilisateur). Si nous devons développer ou faire développer notre propre simulateur, tout se complique, il faudra :

1. Identifier correctement nos besoins en terme de test.
2. Évaluer la charge et bien planifier.
3. Tester l'outil.
4. Prévoir à la première utilisation la possibilité d'anomalie du côté de l'outil.

Quels sont les risques ?

1. Outil non livré en temps et en heure ce qui va retarder la phase de tests.
2. Impossibilité de tester certains scénarios (en particulier les cas d'erreur) car l'outil n'a pas été correctement spécifié.
3. Outil non testé qui va rendre le *debug* très pénible et entraîner des conflits entre développeur et utilisateur de l'outil.
4. Un bon outil va donc faciliter la phase de tests, il faudra donc veiller à sa qualité.

Dans le chapitre automatisation j'aborderai la question des outils plus en détail.

## 3. Conception et codage

Mais le plus efficace est d'intégrer la testabilité lors de la conception. Deux aspects sont à prendre en compte :

1. La contrôlabilité : la possibilité de contrôler l'état d'un composant en vue de le tester.
2. L'observabilité : la possibilité d'observer les résultats de test (intermédiaires et finaux).

### 3.1. Contrôlabilité

Roy Osherove, dans son article "Achieving And Recognizing Testable Software Designs", propose les règles suivantes pour atteindre cet objectif et améliorer la testabilité :

1. Des exécutions partielles : pour déboguer une fonctionnalité nous pouvons exécuter de façon partielle une partie des tests unitaires.
2. Des résultats identiques à chaque exécution : les résultats ne dépendent pas de l'état du système.
3. Pas de configuration : aucune configuration du système n'est nécessaire pour tourner les tests.
4. L'ordre des tests n'est pas important : nous pouvons exécuter les tests dans l'ordre que nous voulons.
5. Temps d'exécution rapide : la masse de tests unitaires est importante, pour accélérer l'exécution, nous évitons par exemple de nous connecter aux bases de données.

Pour respecter ces règles, il faudra mettre en place des règles de codage qui visent à faciliter le test.
Si nous atteignons ces objectifs nous gagnons en "contrôlabilité". Le résultat est la possibilité de tester complètement et simplement le code par morceaux. Un autre point qui améliore la testabilité est de limiter la complexité du code. Il est mesurable par la complexité cyclomatique (lié au nombre d'imbrications). Plus cette mesure est élevée plus le code est source d'erreurs. De plus, il augmente le nombre de jeux d'essais qui permettent le test. D'autre part nous allons limiter le nombre de jeux d'essais et augmenter la contrôlabilité. Là encore, ce sont les règles de codage qui vont améliorer la testabilité.

### 3.2. Observabilité

Voilà nos tests unitaires et d'intégration ont été réalisés et ont permis de déboguer le code, reste à vérifier que notre logiciel a bien le niveau de qualité en terme de fonctionnalités attendues mais aussi en terme de rendu de service (stabilité, sécurité...). Il nous faut passer aux tests système. Là encore, un certain nombre d'éléments doivent être prévus pour nous aider dans nos tests :

1. Politique de gestion d'erreurs : date, type d'erreur sont des informations qui permettront de déboguer et trouver rapidement la cause en cas d'erreur détectée par les tests.
2. Politique de log : différents niveaux que l'on peut activer ou désactiver à chaud. Si ces logs doivent être utilisés par le support, il faudra les rendre compréhensibles et accessibles à tous. Ils facilitent le *debug* et permettent d'analyser les causes d'erreurs.
3. Connaissance des états internes du système ou des objets : donner les moyens de connaître les états du système, voire de les gérer, vont permettre de mieux contrôler et observer l'évolution du système lors des tests et ainsi de vérifier la consistance du système. Cela peut être fait par l'intermédiaire d'une base de données ou d'API.

Certains outils tels que les émulateurs ou les *débogueurs* peuvent être plus ou moins performants en terme de fonctionnalités et vont eux aussi permettre d'améliorer l'observabilité. Dans le développement d'un logiciel embarqué, l'absence d'émulateur peut ralentir considérablement la sortie d'un produit, car le *debug* se fera à tâtons par l'émission d'hypothèses.

### 3.3. Dès la conception

La testabilité doit être prise en compte dès la conception du logiciel, car l'architecture peut améliorer grandement ce critère. Il est d'ailleurs très difficile de l'améliorer a posteriori :

1. Tenter de rajouter des tests unitaires sur un logiciel où cet aspect a été négligé peut se révéler une tâche ardue. Car il est peu probable qu'il ait été conçu de façon modulaire.
2. De même, mettre à niveau des logs ou de la gestion d'erreur peut s'avérer être une tâche coûteuse mais éventuellement à mettre en balance avec le coût du *debug*, support et maintenance ainsi que l'image vis-à-vis du client.

## 4. Plan de tests

Pour écrire un plan de tests complet, nous aurons besoin des points suivants :

1. Les spécifications sont connues et leurs évolutions maîtrisées.
2. Elles sont complètes et abordent tous les aspects (fonctionnels, opérationnels, performance, sécurité, etc. ).

Comment écrire un plan de tests si nous ne savons pas quoi vérifier ? Sans exigences définies, comment faire une couverture et être sûr de la complétude de nos tests ?
Néanmoins, si elles sont absentes une technique de tests peut pallier ce manque : les tests dits exploratoires. Il s'agit d'écrire et d'exécuter les tests en découvrant le logiciel. Ces sessions seront conduites avec des objectifs qualité définis (comme l'usage d'une fonctionnalité, la sécurité...).
Cette technique peut être appliquée si :

1. Le produit est connu (existe par ailleurs, les testeurs connaissent le métier).
2. Les testeurs ne sont pas des débutants.
3. Les conséquences des erreurs sont minimes (données sensibles, contrat, sécurité des personnes).

Néanmoins, il faut rester conscient que certains points peuvent échapper au testeur, car il n'a pas la maîtrise des spécifications de façon complète et sûre. De plus, tout est sérialisé, ce qui augmente les délais de livraison.

## 5. Exécution des tests

### 5.1. Tests Unitaires

Les tests unitaires sont essentiels pour améliorer la

testabilité :

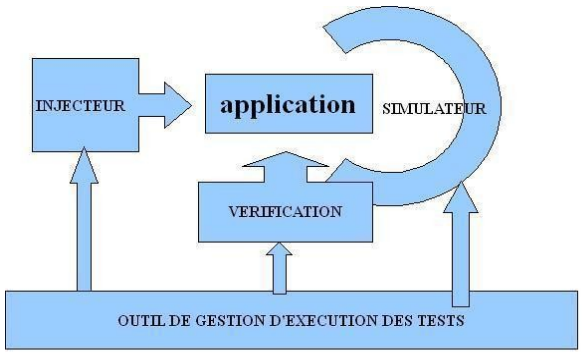
1. Ils vont faciliter le déroulement des tests système : les anomalies liées à des erreurs de codage sont plus facilement détectées et corrigées à ce niveau.
2. Lors d'évolution ou de correction, ils vont permettre de localiser les erreurs de façon fine et permettre de corriger des anomalies ou d'améliorer le code sans crainte de non-régression.

### 5.2. Tests d'intégration

Il est absolument nécessaire de planifier ces tests avec tous les intervenants avant de commencer les tests système. Si cette phase n'est pas réalisée, au lieu de tester les aspects fonctionnels les testeurs feront face à un système qui ne fonctionne pas. Ils devront faire appel à tous les intervenants pour déboguer le système petit à petit. Ils prendront du retard et des conflits vont apparaître.
Une phase d'intégration planifiée et menée à terme facilite les tests système.

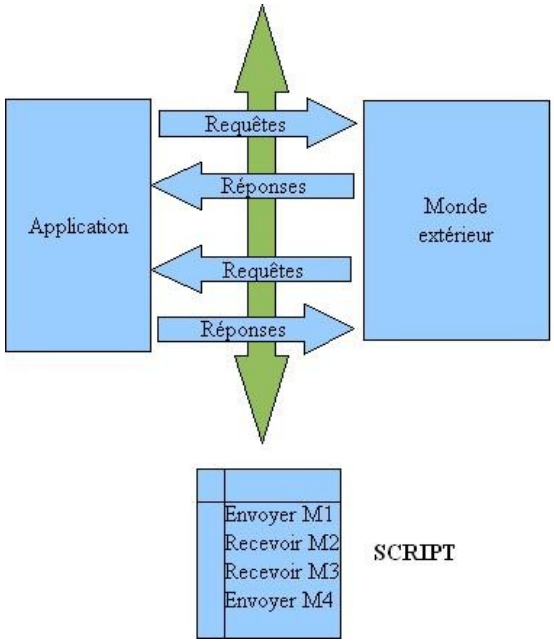
### 5.3. Automatisation

Tester prend du temps. Quand le nombre de tests devient trop important, les versions à tester se multiplient et l'automatisation est la solution qui s'impose. Sinon faute de temps nous allons procéder à des coupes dans les tests pour livrer en temps et en heure et prendre des risques sur la qualité du livrable. L'automatisation est donc nécessaire mais encore faut-il s'y prendre à l'avance. Le schéma suivant résume l'outillage nécessaire et nos choix vont augmenter ou non la testabilité de l'application de par les possibilités de nos outils :




**Injecteur** : l'outillage qui va permettre d'appeler les interfaces externes de l'application comme par exemple des scripts de tests écrits pour des API (Java, C++, Soap) ou des tests automatiques d'interfaces (QTP, WINRUNNER, Selenium...).
**Simulateur** : outil qui simule le monde extérieur. Si l'application communique avec d'autres applications il faut simuler la communication entre ces applications. Si le monde extérieur se réduit aux utilisateurs finaux, il n'y a pas besoin de simulateur on utilise le même outil que l'injecteur, soit des outils d'automatisation de tests d'interfaces.
**Vérification** : toutes les vérifications qui vont déterminer si le test est réussi ou non.
**Outil de gestion d'exécution des tests** : le maître d'oeuvre qui va séquencer les autres outils et centraliser les résultats des tests.

L'injecteur est la partie la moins complexe, sauf pour les tests de performance. Pour ces tests il faudra également prévoir le hardware nécessaire. L'autre difficulté est d'être capable de simuler une utilisation réaliste de l'application.
La partie vérification : pour automatiser il faut pouvoir accéder à certaines données qui sont les sorties, mais parfois à des données internes en base ou à l'état d'un service en cours. Peut-être faudra-t-il faire développer des API internes ?
Les résultats devront être déterministes. Lors d'un de mes projets nous avions réalisé un système d'automatisation de la façon suivante : le test était exécuté manuellement une première fois ; le résultat après vérification était enregistré comme référence ; le test automatisé vérifiait la sortie par rapport à la référence. Malheureusement il s'agissait d'une liste de contacts et à la version suivante l'ordre des contacts n'était plus déterministe. Impossible de repasser les tests automatiques même en changeant les résultats. Nous avons demandé une relivraison car c'était plus facile côté développement.
Autre exemple : pour utiliser Sélénium (outil d'automatisation d'interfaces Web) les objets graphiques doivent être identifiés par un identifiant unique, qui doit être positionné par le développeur d'applications Web.
Côté simulateur il faut qu'il réponde aux besoins de tests, c'est-à-dire être capable de simuler tous les cas de figure, cas normaux et cas d'erreurs. La meilleure solution est de scripter le comportement du monde extérieur.




Ici la conception du simulateur est très simple, elle consiste essentiellement à envoyer et recevoir des messages décrits dans les scripts, gérer des expirations de délais, vérifier que le message reçu est conforme à celui du script et fournir un compte rendu d'exécution du script. Aucune interprétation du message n'est demandée dans ce cas.
La plus mauvaise solution serait de simuler avec du code le comportement du monde extérieur. C'est-à-dire que le simulateur va interpréter les messages qu'il reçoit et répondre en fonction. Dans ce cas le développement du simulateur va être aussi complexe que celui de l'application et poser les problèmes décrits dans le chapitre II-B. Il y a également de fortes chances que l'on ne pourra

pas tester de façon exhaustive les fonctionnements inattendus ou les expirations de délais.
Le choix des outils de manière générale est crucial et peut réduire considérablement le nombre de cas de tests ou les rendre difficilement réalisables, je pense en particulier aux cas d'erreurs.
De même travailler en amont en intégrant lors de la conception des exigences relatives au test, est nécessaire (par exemple des API internes qui permettraient de vérifier des états de transactions) et facilite les activités de test. Cela est aussi vrai pour les activités de maintenance.

## 6. Planification

La livraison des fonctionnalités dans le temps peut également jouer. Par exemple dans le cadre d'un développement itératif, les fonctionnalités de création d'abonnés ont été livrées mais pas celles de suppression. Après un test de création pour détruire les données, il faut à chaque fois arrêter le serveur, détruire en base et redémarrer le serveur tout en se synchronisant avec les personnes utilisant ce même serveur. Compliqué, il eût été plus simple de livrer directement les fonctions de suppression.
De même, planifier la livraison des corrections d'anomalies peut faciliter l'exécution des tests surtout en cas d'anomalie bloquante.

## 7. Conclusion

La testabilité se joue dès les débuts de la réalisation de l'application à toutes les étapes (définition de la stratégie de test, spécifications, conception...). Le test et le développement doivent donc travailler ensemble avec comme objectif rendre l'application testable. Les méthodes agiles facilitent la gestion de cette problématique (écriture des tests avant le codage, participation des utilisateurs finaux, équipe projet mixte etc...) mais je ne saurais en dire plus n'étant pas du tout une spécialiste de ces méthodes.
Tous ces pré-requis sont nécessaires pour tester dans les meilleures conditions :

1. Des objectifs, des risques identifiés orientent nos tests.
2. Sans spécifications, pas de références pour nos oracles de test.
3. Sans réel moyen de test, l'efficacité des tests est diminuée ou bien les tests sont fastidieux à faire, etc.
4. Un logiciel testable va de pair avec une bonne conception basée sur les principes de modularité, etc.
5. Respecter les différentes phases de test (unitaires, intégration et système).
6. Les outils de tests livrés à temps et correctement testés.
7. Des *logs* et une gestion d'erreurs facilitent le *debug*.
8. Planification des livraisons en prenant en compte les besoins des tests.

Évaluer la testabilité et mettre en amont les moyens pour l'améliorer sont donc nécessaires à la bonne exécution de l'activité de test. Cela a un coût, nécessite de la rigueur et du travail en amont, mais accélère les phases de test.

## 8. Références, liens

Achieving And Recognizing Testable Software Designs Part I
Roy Osherove
le forum dédié aux pratiques de test : Lien158
les articles et tutoriels sur les tests : Lien159

*Retrouvez l'article de Dominique Mereaux en ligne : Lien160*

# Liens

Lien2 : http://linsolas.developpez.com/articles/java/outils/builds/
Lien4 : http://www.developpez.net/forums/f1360/java/edi-outils-java/build/autres/
Lien5 : http://www.developpez.net/forums/f1001/general-developpement/conception/outils/integration-continue/
Lien6 : http://zenika.developpez.com/articles/java/build/gradle/
Lien7 : http://www.araneaframework.org/
Lien8 : https://squill.dev.java.net/
Lien9 : http://dow.ngra.de/
Lien10 : http://www.youngprofessionalsforum.cz/
Lien11 : http://www.zeroturnaround.com/jrebel/comparison
Lien12 : http://groups.google.com/group/liverebel-private-beta
Lien13 : http://java.developpez.com/interview/zeroturnaround/jrebel/en/
Lien14 : http://www.zeroturnaround.com/
Lien15 : http://www.zeroturnaround.com/jrebel/current/
Lien16 : http://www.developpez.net/forums/d901582/java/edi-outils-java/zeroturnaround-sort-version-3-0-loutil-productivite-jrebel/
Lien17 : http://www.developpez.net/forums/f199/java/edi-outils-java/
Lien18 : http://java.developpez.com/interview/zeroturnaround/jrebel/
Lien19 : http://developer.android.com/reference/android/app/ListActivity.html
Lien20 : http://mickael-lt.developpez.com/tutoriels/android/personnaliser-listview/fichiers/DVP_ListActivity.zip
Lien21 : http://mickael-lt.developpez.com/tutoriels/android/personnaliser-listview/fichiers/DVP_List1.zip
Lien22 : http://mickael-lt.developpez.com/tutoriels/android/personnaliser-listview/fichiers/DVP_List2.zip
Lien23 : http://mickael-lt.developpez.com/tutoriels/android/personnaliser-listview/fichiers/DVP_List2_event.zip
Lien24 : http://developer.android.com/index.html
Lien25 : http://android.developpez.com/faq/
Lien26 : http://ydisanto.developpez.com/tutoriels/android/debuter/
Lien27 : http://mickael-lt.developpez.com/tutoriels/android/personnaliser-listview/
Lien28 : http://www.ubuntu.com/getubuntu/download-server
Lien29 : http://www.virtualbox.org/
Lien30 : http://pecl.php.net/
Lien31 : http://julien-pauli.developpez.com/tutoriels/php/compilation/#LIII
Lien32 : http://www.php.net/ChangeLog-5.php
Lien33 : http://php.net/svn.php
Lien34 : http://docsun.cites.uiuc.edu/sun_docs/C/solaris_9/SUNWdev/LLM/p12.html#CHAPTER3-1
Lien35 : http://www.iecc.com/linker/
Lien36 : http://svn.php.net/viewvc/pear/ci/phpfarm/trunk/README?view=markup
Lien37 : http://svn.php.net/viewvc/php/php-src/trunk/Zend/README.ZEND_VM?view=markup
Lien38 : http://sebastian-bergmann.de/archives/504-PHP-5.1-Performance.html
Lien39 : http://julien-pauli.developpez.com/tutoriels/php/apc/
Lien40 : http://pear.php.net/manual/en/guide.users.commandline.cli.php
Lien41 : http://svn.xdebug.org/cgi-bin/viewvc.cgi/vld/?root=php
Lien42 : http://www.bytekit.org/
Lien43 : http://github.com/sebastianbergmann/bytekit-cli
Lien44 : http://www.imagemagick.org/
Lien45 : http://www.php.net/imagick
Lien46 : http://memcached.org/
Lien47 : http://www.php.net/memcache
Lien48 : http://www.php.net/scream
Lien49 : http://www.php.net/bbcode
Lien50 : http://www.php.net/parsekit
Lien51 : http://pyyaml.org/wiki/LibYAML
Lien52 : http://www.php.net/yaml
Lien53 : http://www.php.net/http
Lien54 : http://libharu.org/wiki/Main_Page
Lien55 : http://devzone.zend.com/article/4044
Lien56 : http://www.php.net/haru
Lien57 : http://www.teslacore.it/wiki/index.php?title=AMFEXT
Lien58 : http://www.php.net/book.svn
Lien59 : http://www.xmailserver.org/xdiff.html
Lien60 : http://www.php.net/xdiff
Lien61 : http://sphinxsearch.com/
Lien62 : http://www.php.net/sphinx
Lien63 : http://www.php.net/runkit
Lien64 : http://www.xdebug.org/
Lien65 : http://pecl.php.net/package/docblock
Lien66 : http://www.tenouk.com/ModuleW.html
Lien67 : http://www.cs.bu.edu/teaching/cpp/writing-makefiles/
Lien68 : http://www.iecc.com/linker/
Lien69 : http://melem.developpez.com/tutoriels/langage-c/compilation-separee/
Lien70 : http://julien-pauli.developpez.com/tutoriels/php/compilation/
Lien71 : http://www.admixweb.com/2009/06/23/how-to-easily-create-a-javascript-framework-part-3/
Lien72 : http://kalyparker.developpez.com/articles/js/VOZ-partie-2/
Lien73 : http://kalyparker.developpez.com/articles/js/VOZ-partie-3/fichiers/vozpart3_fr.html
Lien74 : http://kalyparker.developpez.com/articles/js/VOZ-partie-1/
Lien75 : http://kalyparker.developpez.com/articles/js/VOZ-partie-3/
Lien76 : http://loic-joly.developpez.com/articles/interview-james-reinders/
Lien77 : http://loic-joly.developpez.com/articles/intel-software-conference-2010/
Lien78 : http://www.intel.com/idf/
Lien79 : http://loic-joly.developpez.com/articles/interview-james-reinders-2010/

Lien80 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n2118.html
Lien81 : http://www.open-std.org/Jtc1/sc22/wg21/docs/papers/2009/n2844.html
Lien82 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2007/n2439.htm
Lien83 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2004/n1610.html
Lien84 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2004/n1720.html
Lien85 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n1984.pdf
Lien86 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1757.html
Lien87 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2008/n2546.htm
Lien88 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2008/n2541.htm
Lien89 : http://www.open-std.org/Jtc1/sc22/wg21/docs/papers/2009/n2927.pdf
Lien90 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2007/n2343.pdf
Lien91 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1757.html
Lien92 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n1987.htm
Lien93 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2007/n2431.pdf
Lien94 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2007/n2347.pdf
Lien95 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2008/n2764.pdf
Lien96 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1791.pdf
Lien97 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2008/n2657.htm
Lien98 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2007/n2179.html
Lien99 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2008/n2659.htm
Lien100 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2007/n2340.htm
Lien101 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2004/n1653.htm
Lien102 : http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1811.pdf
Lien103 : http://farscape.developpez.com/Articles/TechDays2008_1/
Lien104 : http://farscape.developpez.com/Articles/VisualStudio2010/
Lien105 : http://projets.developpez.com/projects/qextend
Lien106 : http://www.developpez.net/forums/d906429/c-cpp/bibliotheques/qt/naissance-projet-qextend/
Lien107 : http://qt.nokia.com/products/appdev/add-on-products/catalog/4/new-qt-apis/mobility
Lien108 : http://www.forum.nokia.com/Tools_Docs_and_Code/Tools/IDEs/Nokia_Qt_SDK/
Lien109 : http://qt.nokia.com/files/pdf/qt-mobility-whitepaper-1.0.0
Lien110 : http://qt.nokia.com/
Lien111 : http://www.developpez.net/forums/d914761/c-cpp/bibliotheques/qt/sortie-qt-mobility-1-0-0-a/
Lien112 : http://qt.nokia.com/doc/qq/32/qq32-next-gen-uis.html
Lien113 : http://doc.trolltech.com/qq/
Lien114 : http://doc.qt.nokia.com/4.6/graphicsview.html
Lien115 : http://doc.qt.nokia.com/4.6/animation-overview.html
Lien116 : http://doc.qt.nokia.com/4.6/statemachine-api.html
Lien117 : http://doc.qt.nokia.com/4.6/qgraphicseffect.html
Lien118 : http://doc.qt.nokia.com/4.6/qgraphicsblureffect.html
Lien119 : http://doc.qt.nokia.com/4.6/qgraphicsblureffect.html#blurRadius-prop
Lien120 : http://doc.qt.nokia.com/4.6/qgraphicsitem.html#setBlurEffect
Lien121 : http://doc.qt.nokia.com/4.6/qgraphicsitemgroup.html
Lien122 : http://doc.qt.nokia.com/4.6/animation-overview.html
Lien123 : http://doc.qt.nokia.com/4.6/qobject.html
Lien124 : http://doc.qt.nokia.com/4.6/qabstractanimation.html
Lien125 : http://doc.qt.nokia.com/4.6/qpropertyanimation.html
Lien126 : http://doc.qt.nokia.com/4.6/qobjects.html
Lien127 : http://doc.qt.nokia.com/4.6/qvariantanimation.html#startValue-prop
Lien128 : http://doc.qt.nokia.com/4.6/qvariantanimation.html#endValue-prop
Lien129 : http://doc.qt.nokia.com/4.6/qvariantanimation.html#setKeyValueAt
Lien130 : http://doc.qt.nokia.com/4.6/qpushbutton.html
Lien131 : http://doc.qt.nokia.com/4.6/qabstractanimation.html#DeletionPolicy-enum
Lien132 : http://doc.qt.nokia.com/4.6/qabstactgraphicsitem.html
Lien133 : http://doc.qt.nokia.com/4.6/qgraphicsitem.html
Lien134 : http://doc.qt.nokia.com/4.6/qgraphicsrectitem.html
Lien135 : http://doc.qt.nokia.com/4.6/qobject.html#Q_PROPERTY
Lien136 : http://doc.qt.nokia.com/4.6/qpropertyanimation.html
Lien137 : http://doc.qt.nokia.com/4.6/statemachine-api.html
Lien138 : http://doc.qt.nokia.com/4.6/qstate.html#assignProperty
Lien139 : http://doc.qt.nokia.com/4.6/qvariant.html
Lien140 : http://doc.qt.nokia.com/4.6/qabstracttransition.html
Lien141 : http://doc.qt.nokia.com/4.6/qeventtransition.html
Lien142 : http://doc.qt.nokia.com/4.6/qkeyeventtransition.html
Lien143 : http://doc.qt.nokia.com/4.6/qmouseeventtransition.html
Lien144 : http://doc.qt.nokia.com/4.6/qsignaltransition.html
Lien145 : http://doc.qt.nokia.com/4.6/qstatemachine.html
Lien146 : http://qt.nokia.com/doc/4.6/qstate.html#assignProperty
Lien147 : http://doc.qt.nokia.com/4.6/qwidget.html#visible-prop
Lien148 : http://doc.qt.nokia.com/4.6/qeasingcurve.html#Type-enum
Lien149 : http://doc.qt.nokia.com/4.6/qstatemachine.html
Lien150 : http://qt-quarterly.developpez.com/qq-32/prochaine-generation-ui/fichiers/qq32-blurstates.zip
Lien151 : http://qt-quarterly.developpez.com/qq-32/prochaine-generation-ui/
Lien152 : http://mac.developpez.com/livres/
Lien153 : ftp://ftp-developpez.com/baptiste-wicht/tutoriels/java/tests/mocks/easymock/fichiers/source.zip
Lien154 : http://fchabanois.developpez.com/tutorial/java/jmockit/
Lien155 : http://www.developpez.net/forums/f413/java/edi-outils-java/tests-performance/
Lien156 : http://easymock.org/EasyMock2_5_2_Documentation_fr.html
Lien157 : http://baptiste-wicht.developpez.com/tutoriels/java/tests/mocks/easymock/
Lien158 : http://www.developpez.net/forums/f623/general-developpement/conception/methodes/gestion-projet/test/
Lien159 : http://conception.developpez.com/cours/?page=qualite-cat#tests
Lien160 : http://dominique-mereaux.developpez.com/tutoriels/general/ameliorer-testabilite/