Spring for Android

Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES en informatique de Gestion

par :

Yuri Allendes

Conseiller au travail de Bachelor :

Peter DAEHNE, professeur HES

Genève, 16.11.2012

Haute École de Gestion de Genève (HEG-GE)

Filière Informatique de gestion

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre de Bachelor en Informatique de Gestion. L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Genève le 16.11.2012

Yuri Allendes

Remerciements

Je souhaiterais tout d'abord remercier mon professeur à la Haute Ecole de Gestion de Genève, M. Peter DAEHNE pour son encadrement, son soutien et ses conseils avisés qui m'ont permis de réaliser ce mémoire.

Je voudrais également remercier mon meilleur ami Jonathan Baeriswyl, pour m'avoir permis d'utiliser sa connexion internet dans l'enceinte de l'EPFL.

Enfin j'aimerais remercier ma famille qui m'a soutenu durant toute la durée de mes études.

Résumé

Spring est l'un des frameworks Java le plus utilisé. Sa popularité est certainement due au fait qu'il soit très complet, et propose diverses extensions du framework principal.

Ces extensions, qui sont des frameworks à part entière, sont pour la plupart indépendantes l'une de l'autre, cela permet de ne pas alourdir le projet avec des fonctionnalités non utilisées.

Avec la nouvelle extension « Spring for Android », sortie en 2012, Spring vient étendre sa gamme de frameworks.

Le but de ce mémoire est de découvrir ce nouveau framework qu'est « Spring for Android » et d'analyser le client REST proposé par ce framework.

Pour cela nous commencerons par un historique de Spring afin de mieux comprendre son fonctionnement principal, puis nous passerons en revue les différentes extensions que « Spring for Android » propose.

Nous étudierons aussi l'importance qu'il peut y avoir à développer une application sur Android.

Enfin nous parcourrons une à une les fonctionnalités disponibles sur le framework, et en mettrons en pratique quelques-unes par un exemple concret d'application Android.

Mots clés:

Android, Application Mobile, Client REST, Framework, JSON, Maven, Spring, Spring for Android.

Table des matières

Liste des Tableaux

Liste des Figures

Introduction

Avec la montée en popularité des Smartphones l'utilisation du système d'exploitation Android ne fait qu'accroitre et les applications dédiées à l'OS commencent à fleurir. Afin de se distinguer des autres applications, les projets réalisés aujourd'hui sont d'une grande complexité.

De façon à simplifier la réalisation et d'écourter les temps de développement il est conseillé d'utiliser un framework. Malheureusement, bien que les applications Android soient basées sur le langage Java, les frameworks pour ce langage ne sont pas compatible avec Android. Toutefois, il existe des framework spécialement conçu pour Android, mais ceux-ci étant récents ils ne sont, pour la plupart, pas très matures.

Spring, qui est un framework très connu dans le monde Java, et qui a fait ses preuves, sort une nouvelle extension dédiée spécialement à Android : « Spring for Android ».

Cette extension de Spring adapte certaines fonctionnalités déjà présentes sur d'autres modules Spring, tel qu'un client REST, ressemblant au client REST présent dans « Spring Data REST ».

Ce mémoire est destiné aux développeurs désirant en savoir plus sur l'utilité d'une application Android et des avantages liés à l'utilisation d'un framework tel que « Spring for Android »

Pour cela, dans un premier temps, nous définirons ce qu'est un framework et nous passerons en revue certains des plus connus, puis nous étudierons le framework Spring avant de nous lancer sur l'énumération de ses extensions.

Nous nous familiariserons ensuite avec les concepts et technologies qui seront employées lors de l'analyse du framework « Spring for Android »

Nous examinerons enfin les différentes fonctionnalités proposées par cette extension et les mettrons en pratique dans un exemple concret.

1. Framework

1.1 ***Définition***

Un framework, (cadre de travail) est un ensemble normalisé de concepts, de pratiques et de critères qui servent de référence afin de résoudre des problèmes de nature similaire ; il fournit dans le cadre du développement de logiciels, une structure de support dans laquelle un autre projet (logiciel) peut être organisé et développé.

Un framework contient généralement

- une ou plusieurs bibliothèques
- un guide architectural

Les applications étant devenues de plus en plus riches et complexes, l'utilisation d'un framework permet d'en faciliter le développement. Les applications restent ainsi bien structurées et leurs temps de développement en est réduit. Ceci a pour effet de baisser les coûts de construction ainsi que les coûts de maintenance, car l'application étant bien structurée, elle est aussi plus facilement maintenable.

Malgré tous les avantages d'un framework, Il faut être attentif au temps de prise en main par les développeurs qui peut s'avérer assez long selon la complexité du framework utilisé ; il faut aussi être attentif aux évolutions du framework (mises à jour) qui peuvent amener à modifier les applications existantes mais qui sont souvent nécessaires d'un point de vue sécurité.

1.2 ***Différents framework JAVA et Android***

Il existe une multitude de frameworks pour le langage java, en voici quelques-uns :

APACHE TAPESTRY 5

Framework créé par Howard Lewis Shipe, qui fait maintenant partie de la fondation Apache et qui sert à créer des applications web Java basée sur Java EE. Tapestry utilisait des pages XML afin d'implémenter les applications web, mais depuis la version 5, il n'utilise plus que les annotations.

Ce framework est libre de droit et il en est actuellement à sa version 5.3.6.

Lien : http://tapestry.apache.org

LEONARDI

Framework open source qui permet de créer rapidement des interfaces homme-machine complètes (MMI ou Man-Machine Interface).

Leonardi existe en version open source (Leonardi free) ainsi qu'en version commerciale (business first), la version actuelle est la version 8.8

Lien : http://www.leonardi-free.org

PLAY FRAMEWORK

Framework open source créé par Guillaume Bort qui permet de réaliser facilement des applications web en java ou en Scala ; il a l'avantage d'être rapide d'emploi (ne nécessite presque pas de configuration). Sa version actuelle est la 2.0.4

Lien : http://www.playframework.org

STRUTS 2

Framework open source succédant à Struts, qui permet de créer des applications Web java en utilisant le pattern Modèle-Vue-Contrôleur (MVC). Il permet de créer des applications REST conventionnelles, mais aussi de mettre en œuvre des technologies comme SOAP ou AJAX. C'est un des principaux concurrents de Spring MVC (module Spring permettant de construire des applications Web)

Il en est actuellement à sa version 2.3.4.1

Lien : http://struts.apache.org/2.x/

VAADIN

Framework open source pour la création d'applications internet riche (RIA)[1] coté serveur. Il ne nécessite aucun plugin du coté client, et il a la particularité d'utiliser uniquement JAVA comme langage de programmation (pas de JavaScript, html, CSS). Vaadin est basé sur Google Web Toolkit (GWT) mais se

[1] Les RIA sont des applications Web qui fonctionnent comme des applications de bureau traditionnelles (logiciel), mais à l'instar des applications de bureau les RIA sont accessibles uniquement avec un navigateur web, il n'est donc pas nécessaire d'installer le logiciel sur son ordinateur. Néanmoins une application RIA peut nécessiter l'installation de plugins sur son ordinateur tels que ActiveX ou Flash.

différencie de celui-ci car il ne nécessite pas de compilation pré-déploiement et qu'il est uniquement basé du coté serveur. La version actuelle est la 6.8.5

Lien : https://vaadin.com/home

ROBOGUICE

Framework Android qui est une extension du framework google-guice pour l'injection de dépendance. Il permet d'éliminer du code répétitif en donnant la possibilité d'injecter directement des `View`, des `Ressources` et des `System Service` dans les `Activity`.

Il en est à sa version 2.0

Lien : http://code.google.com/p/roboguice

ANDROID BINDING

Framework open source pour Android qui aide à implémenter le pattern MVC [2] ou le pattern MVVM[3] et permet donc de séparer la partie présentation de la partie fonctionnelle. Les Activity ne se chargent plus que de faire le lien en la vue et le modèle. Le seul point contraignant est que l'édition des View intégré au plugin Eclipse ne fonctionne plus avec ce framework. Il n'en est, pour l'instant, qu'à la version 0.52

Lien : http://code.google.com/p/android-binding/

RESLET

Framework open source reposant sur les concepts REST, il propose à la fois un coté client et un coté serveur, les deux reposant sur le même API. La particularité est que ce framework existe sur différentes plateformes dont Java SE, Java EE, mais aussi Android. Il est donc un concurrent direct du framework « Spring for Android »

Il en est à la version 2.0.15.

Lien : http://www.restlet.org

---

2 Model View Controler (Modèle Vue Contrôleur)

3 Model View ViewModel

2. Spring

2.1 ***Historique***

M. Rod Johnson, auteur du livre « Expert One-on-One J2EE Design and Development », explique qu'une application J2EE bien conçue devrait :

- Etre Robuste
- Etre performante et évolutive
- Tirer parti des principes de conception Orientés Objet
- Eviter toute complexité inutile
- Etre maintenable et extensible
- Etre livrée à temps
- Etre facile à tester
- Promouvoir la réutilisation

Il a créé le framework Spring afin d'aider à respecter ces principes. La première version a été publiée le 24 mars 2004 sous la licence Apache 2.0 ; elle ne contenait qu'une seule librairie et permettait notamment d'accéder aux informations hébergées dans une base de données relationnelles à l'aide de solutions ORM (Object Relational Mapping) telles que Hibernate.

La première version de ce framework va rapidement évoluer en proposant de nombreuses fonctionnalités nouvelles. Aujourd'hui Spring, qui a été racheté par VMWare en 2009, a largement débordé de son cadre initial. Spring framework en est à sa version 3.1.2, mais le framework propose aussi un grand nombre de modules logiciels, et dispose également d'une très grande communauté.

2.2 ***Qu'est-ce que Spring ?***

Spring est décrit comme un framework léger (lightweight framework) pour construire des applications Java, mais contrairement à d'autres frameworks JAVA il permet de construire n'importe quelle sorte d'application JAVA, et pas seulement des applications web.

Spring framework est basé sur le principe d'inversion de contrôle (IoC) et l'injection de dépendance

2.2.1 L'inversion de contrôle

La collaboration des objets les uns avec les autres est un des principes de base de la programmation orientée objet. Cette collaboration est possible uniquement par la

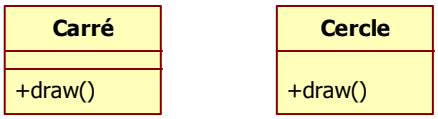
connaissance qu'un objet a d'un autre. Si un objet X a besoin de collaborer avec un objet Y, il est nécessaire que l'objet X connaisse l'identité de Y.

Le problème est que cela crée une dépendance entre les objets X et Y.

Afin de minimiser la dépendance, on peut utiliser un design pattern tel que factory qui va apporter un intermédiaire qui se chargera d'instancier la classe.

2.2.1.1 Exemple

Prenons l'exemple de deux figures géométriques un carré et un cercle qui ont tous les deux une méthode `draw()` qui a pour fonction de dessiner la forme.



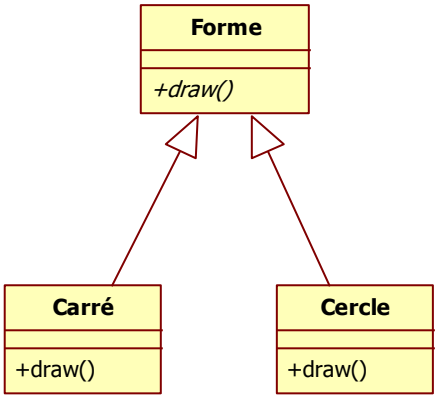
Si l'on veut dessiner ces deux formes on va procéder comme ceci :

MonApplication

```
Carré unCarré = new Carré();
unCarré.draw();
Cercle unCercle = new Cercle();
unCercle.draw();
```

Cette application va devoir connaître la classe `Carré` ainsi que la classe `Cercle` ; il y aura donc une dépendance envers ces deux classes.

Comme nous utilisons un langage Orienté Objet on veut pouvoir utiliser le polymorphisme. Pour cela nous allons utiliser une classe mère `Forme` (qui sera une classe abstraite ou une interface ou alors elle contiendra une méthode abstraite). Cette classe va contenir une méthode `draw()`, qui elle ne contiendra aucun code.

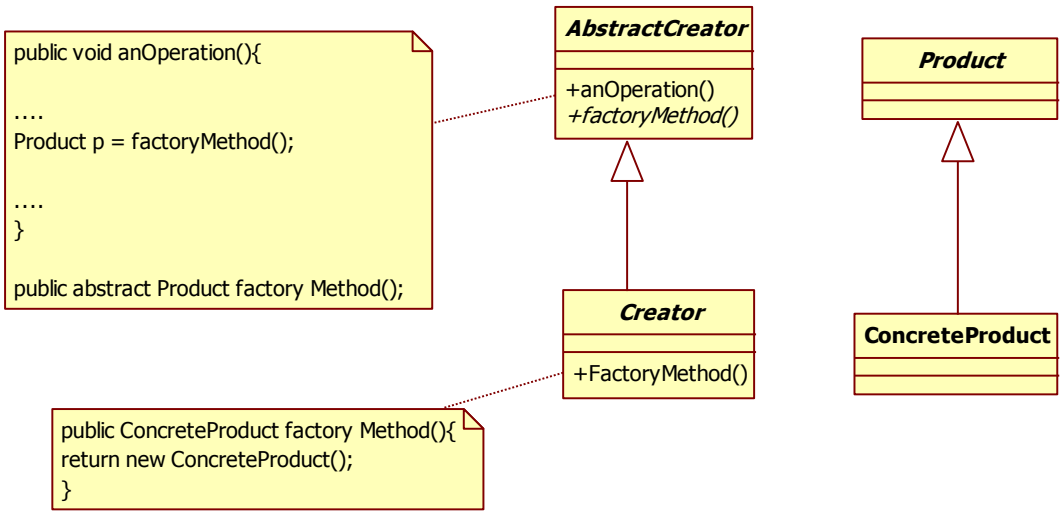


Lorsque l'on va vouloir dessiner un carré on ne va plus appeler directement la méthode `draw()` de la classe `Carré` mais, on va appeler la méthode `draw()` de `Forme` qui ira chercher le code de `draw()` de la classe carré par liaison dynamique :

MonApplication

```
Forme forme = new Carré();
forme.draw();
Forme forme = new Cercle();
forme.draw()
```

Avec cet exemple, bien que l'on utilise le polymorphisme, on ne règle pas le problème de dépendance ; pour cela il va falloir que nous utilisions le design pattern « Factory Method »



En appliquant ce pattern on va pouvoir isoler l'instanciation de l'objet dépendant de l'initialisation

Drawing

```
public class Drawing {
	private Forme forme;

	public setForme(Forme forme){
		this.forme = forme;
	}

	public dessinerForme(){
		this.forme.draw();
	}

}
```

Class

```
Carré unCarré = new Carré();
drawing.setForme(unCarré);

drawing.dessinerForme();
```

L'inversion de contrôle (Inversion of Control, IoC), fonctionne un peu de la même façon. « C'est un patron d'architecture commun à tous les frameworks. Il fonctionne selon le principe que le flot d'exécution d'un logiciel n'est plus sous le contrôle direct de l'application elle-même mais du framework ou de la couche logicielle sous-jacente.

L'inversion de contrôle est un terme générique. Selon la problématique, il existe différentes formes, ou représentation d'IoC. Le plus connu étant l'injection de dépendances. » Source : Wikipédia

2.2.2 Injection de dépendance

Spring, avec l'injection de dépendance, offre un autre moyen d'initialiser les dépendances entre les différentes classes. Au lieu d'implémenter une factory, avec Spring, on aura un fichier de paramétrage au format XML, qui jouera le rôle d'une factory générique totalement paramétrable.

2.2.2.1 Exemple

Si nous adaptons notre exemple à l'injection de dépendance façon Spring, on va créer un fichier XML qui va jouer le rôle de notre factory. Dans celui-ci nous déclarons un identifiant dans « id » et le lien vers notre objet (package + nom de la classe) dans `class`.

La première étape est de déclarer notre factory dans notre application, en indiquant le nom de notre fichier XML (il est possible d'utiliser `ApplicationContext` au lieu de `BeanFactory`, ce qui permet d'utiliser des fonctionnalités plus poussées mais pour notre exemple nous n'en avons pas besoin). Nous allons ensuite instancier `Carré` non pas en faisant `Carré carré = new Carré();` mais en appelant notre factory.

Carré

```
public class Carré {
	public void dessinerForme(){
	//Implémentation du code
	}

}
```

Spring.xml

```
<bean id="carré"
class="NomduPackage.Carré"/>
```

Mon application

```
BeanFactory factory = new ClassPathXmlApplicationContext("Spring.xml");
Carré carré = (Carré) factory.getBean("carré");

carré.dessinerForme();
```

3. Les différentes extensions de Spring

Le framework Spring a beaucoup évolué depuis son lancement. Si bien qu'aujourd'hui, il sert de base à plusieurs autres projets open source faisant partie de la communauté SpringSource.

3.1 ***Spring Security***

Spring Security est un framework qui assure l'authentification, la gestion des autorisations ainsi que d'autres dispositifs de sécurité. Il a été lancé en mars 2004 sous le nom de « Acegi Security » puis, par la suite il a été incorporé aux modules de Spring en 2008, sous son nom actuel.

Spring Security supporte différents modes d'authentification tels que basic[4], digest[5] et par formulaire[6]. Pour ce qui est de la gestion des habilitations, il supporte les mécanismes liés aux rôles utilisateurs.

Aujourd'hui Spring Security est utilisé pour sécuriser les environnements les plus exigeants, tel que les agences gouvernementales, les applications militaires et les banques centrales.

La version actuelle est Spring Security 3.1.3.

3.2 ***Spring Social***

L'extension Spring Social permet de connecter plus facilement son application avec des fournisseurs de « logiciel en tant que service » (Software as a Service, SaaS) tels que

4 HTTP Basic authentification, qui est basé sur un nom d'utilisateur et mot de passe, est le mécanisme d'authentification défini dans la spécification http, Ce mode n'est pas sécurisé, le mot de passe n'étant pas crypté. C'est pourquoi il est préférable de l'utiliser avec un mécanisme de transport sécurisé (HTTPS).

5 HTTP Digest authentification est basée sur un nom d'utilisateur et un mot de passe. Cependant, l'authentification est effectuée par transmission du mot de passe sous une forme cryptée.

6 L'authentification par formulaire également appelé authentification par cookie permet aux développeurs de contrôler l'aspect de l'écran de connexion, mais tout comme le Basic authentification il n'est pas sécurisé.

Facebook ou Twitter. Les applications d'aujourd'hui étant de plus en plus sociales, ce module prend toute son importance.

Afin d'évoluer plus facilement, les modules des fournisseurs ne sont plus inclus dans Spring Social, mais sont indépendants dans leur propre projet avec un module commun qui est le cœur de l'extension (Spring Core), cela permet notamment de réagir plus rapidement lorsque l'API d'un fournisseur change, car il ne faut plus attendre une nouvelle version de Spring Social.

Les principaux modules sont :

- Spring Social Twitter
- Spring Social Facebook
- Spring Social LinkedIn
- Spring Social Tripit
- Spring Social Github

Mais il existe aussi d'autres projets développés par la communauté Spring qui sont appelés Community-Led Projects parmi lesquels on retrouve des modules pour beaucoup d'autres SaaS tels que Dropbox, Flickr, Tumblr, 500px, etc..

La majorité de ces fournisseurs SaaS ont des API REST, qui varient selon le fournisseur, avec de différents concepts, des formats de sortie différent (JSON, XML,…) ainsi qu'une gestion des erreurs différentes. La plupart d'entre eux sécurisent leurs API avec OAuth[7], mais tous n'ont pas la même version (OAuth1.0, OAuth1.0a, OAuth2.0 Working Draft) et comme les versions fonctionnent différemment et que la version 2.0 n'est pas finie et qu'elle est continuellement en train de changer, il devient difficile d'utiliser l'API. Pire encore si le fournisseur SaaS change sa version de OAuth, l'application créée (sans Spring social) avant le changement de version ne fonctionnera plus, d'où l'utilité de Spring Social.

La version actuelle est Spring Social 1.0.2.

7 OAuth est un protocole libre qui permet l'authentification à une API sécurisée d'une façon simple et standard. Pour les développeurs d'une application accédant à une API, OAuth est une façon de publier et d'interagir avec des données protégées. Pour les développeurs fournissant une API, OAuth permet de donner accès aux données tout en protégeant le pseudonyme et le mot de passe des utilisateurs. Source Wikipedia : http://fr.wikipedia.org/wiki/OAuth

3.3 ***Spring Data***

Spring Data rend plus facile la construction d'applications qui utilisent de nouvelles technologies d'accès aux données, telles que les bases de données non relationnelles ou les services de données sur le cloud. Il permet aussi un meilleur soutien des bases de données relationnelles.

Spring Data n'est pas un framework en lui-même mais plutôt un projet open source qui contient plusieurs sous projets spécifiques à une base de données, ou à un service de données ; tous les sous projets peuvent être installés indépendamment. Parmi les plus connus on trouve :

- **JPA** : pour faciliter la création de JPA[8]
- **JDBC Extensions** : support pour les applications utilisant JDBC [9]
- **REST** : fournit un client REST

3.4 ***Spring Mobile***

Spring Mobile est une extension de Spring MVC qui permet de simplifier le développement d'applications web mobiles (non natives). Ce module permet de détecter (du côté serveur) si la requête entrante provient d'un appareil mobile ou non ; il permet aussi de laisser le choix à l'utilisateur de switcher entre la version mobile et la version normale du site internet.

Spring Mobile est une des dernières extensions du framework à être sortie ; la première version stable, Spring Mobile 1.0.0 est sortie peu avant la première version de Spring for Android, et c'est la version actuelle.

---

8 La Java Persistence API est une interface de programmation Java permettant aux développeurs d'organiser des données relationnelles dans des applications utilisant la plateforme Java., Source Wikipedia : http://fr.wikipedia.org/wiki/Java_Persistence_API

9 Java DataBase Connectivity

3.5 ***Spring for Android***

Spring Android est une extension du framework Spring qui vise à simplifier le développement d'applications natives sur Android. C'est le dernier module à être sorti dans la galaxie Spring.

Annoncé fin 2010, le Module aura connu plusieurs versions beta :

- V. 1.0.0 M1 : cette première version comprenait un client REST pour Android.
- V. 1.0.0 M2 : les principales nouveautés de cette version sont l'intégration avec la librairie Jackson pour le marshaling JSON[10], le marshaling XML et le support des flux RSS et des flux Atom.
- V. 1.0.0 M3 : cette version améliore le client REST, et supporte l'intégration de Spring Social.
- V. 1.0.0 M4 : cette version améliore les fonctions disponibles précédemment et supporte l'intégration de Spring Security ainsi qu'avec la librairie Gson qui permet tout comme la librairie Jackson de faire du marshaling JSON ; elle supporte aussi la compression gzip (pour le client REST).
- V. 1.0.0 RC1 : cette version apporte beaucoup d'améliorations. Outre le fait de supporter les dernières versions de Spring Social et Spring Security, d'améliorer le support de compression gzip et de corriger des bugs, elle inclut le support du Basic Authentification, elle ajoute une méthode AssetRessource afin d'accéder aux ressources statiques qui sont stockées dans le répertoire actif d'un projet Android.
- V. 1.0.0 : c'est la version finale sortie le 30 mai 2012, elle n'apporte pas beaucoup de nouveautés, mais elle corrige plusieurs bugs présents dans la version RC1.

10 Opération de transformation d'un objet Java en fichier JSON

4. Android

Android est un système d'exploitation open source, pour appareil mobile (Smartphone, tablette,..), basé sur un Kernel Linux et développé par Google™.

4.1 ***Historique***

En 2003, une startup spécialisée dans le développement d'applications pour téléphone mobile du nom d'Android Inc. est créé par quatre hommes Rich Miner, Andy Rubin, Nick Sears et Chris White.

En 2005, Google™ acquiert la startup et commence secrètement, avec Andy Rubin en tête du projet, le développement d'un OS mobile basé sur le kernel linux.

Le 5 novembre 2007, la création de l'Open Handset Alliance (OHA), un consortium de 34 compagnies (84 compagnies en 2012) du secteur technologique et mobile qui a pour but d'accélérer l'innovation dans la téléphonie mobile, est officiellement annoncé, à l'initiative de Google™. Leur intention est de développer des standards open source pour appareil mobile ; ce même jour Android est annoncé.

Parmi les membres fédérateurs se trouvent :

- des constructeurs de mobiles
- des opérateurs mobiles
- des constructeurs de semi-conducteurs
- des sociétés de services
- des sociétés logicielles

Le 22 octobre 2008 sort le premier téléphone sous Android : le HTC dream, aussi appelé HTC G1.

4.2 ***Quelques chiffres***

Aujourd'hui Android est devenue l'OS mobile le plus utilisé au monde, dépassant iOS d'Apple® avec [11]:

- 1,3 millions de dispositifs activés chaque jour
- Plus de 500 millions de dispositifs activés dans le monde jusqu'à aujourd'hui
- Plus de 25 milliards d'applications téléchargées jusqu'à aujourd'hui

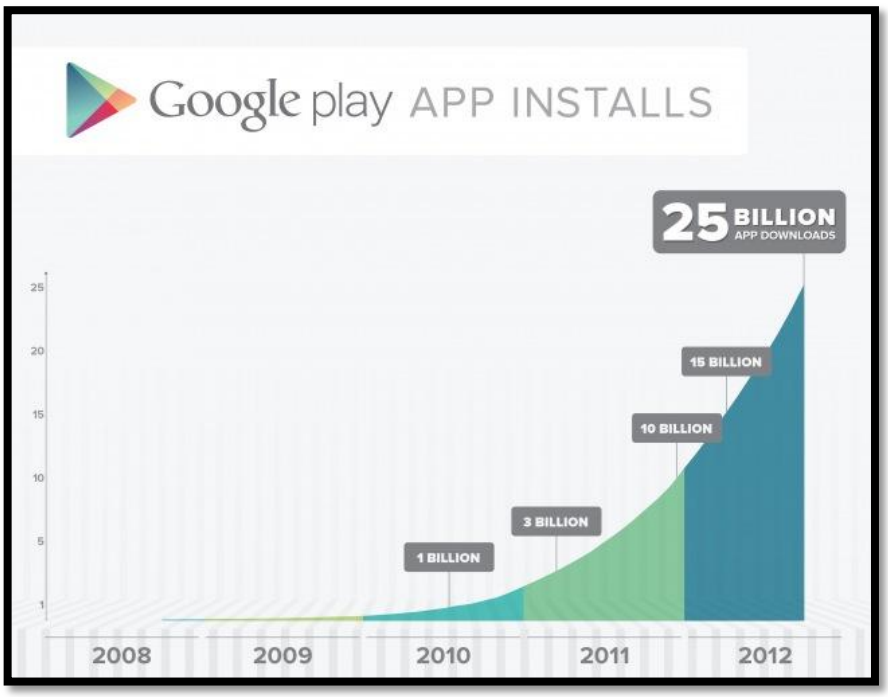
Ces chiffres ne cessent d'augmenter de façon exponentielle.

---

11 Chiffres datant de septembre 2012

Figure 1

Nombre de dispositifs activés sur Google Play

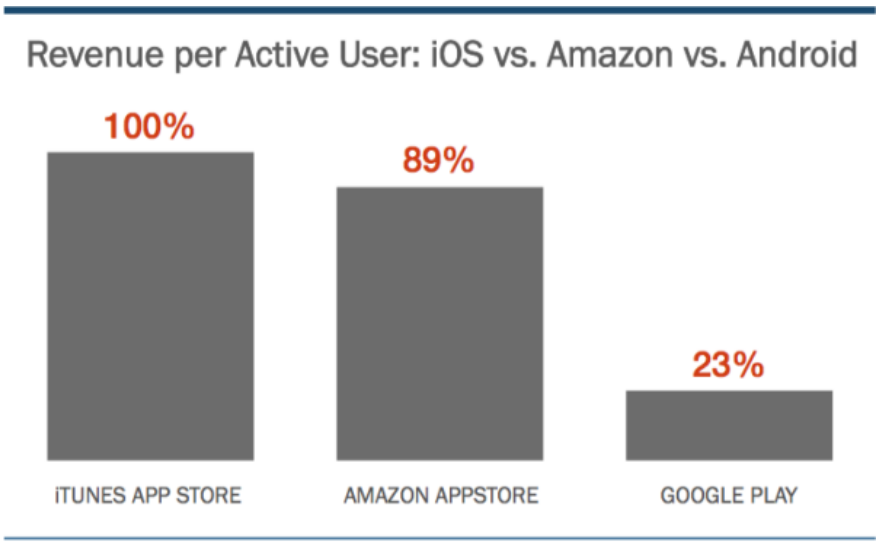


Source Official Android Blog (26 septembre 2012)

Il est donc devenu, en théorie, très intéressant de programmer des applications pour cette plateforme et de les rendre disponibles sur le Play Store (market Android). Pourtant, plusieurs études démontrent que pour l'instant, le développement sur Android ne génère pas assez de revenu. Le cabinet d'analyse Flurry a comparé le revenu de trois différentes plateformes. Pour cela ils ont mesuré le revenu générés par les achats de différentes applications présentes sur les trois Markets et cela sur une période de 45 jours à partir de mi-janvier jusqu'à la fin de Février 2012.

Figure 2

Comparatif des revenus de trois différents Market



Source Flurry Analytics (Février 2012)

Le graphique ci-dessus compare les revenus générés par l'utilisateur à travers des Markets d'iOS, d'Amazon et d'Android, en mettant comme référence l'App Store d'iTunes à 100% et en les comparants aux deux autres.

Néanmoins, cette étude ne prend pas en compte les frais engendrés par la publication des applications (frais développeur plus cher sur l'App Store que sur Google Play), ni du fait qu'il existe d'autres façons pour une application d'être rentable :

- **In app purchase** : Il s'agit de proposer des achats via l'application c'est une sorte de market à l'intérieur de l'application proposant du contenu supplémentaire (bonus, monnaie virtuelle). Les applications proposant ce genre de contenu ont connu un fort essor ces dernières années, notamment parmi les jeux vidéo ; ces applications sont habituellement gratuites et proposent l'intégralité du jeu ou du service.

- **La publicité** : C'est l'un des business model les plus pratiqués mais, bien que la publicité sous forme de bannière puisse être rentable, il faut pour cela que l'application soit téléchargée des millions de fois, car pour générer du revenu, il faut que l'utilisateur clique sur la bannière.

- **Le sponsoring** : Bien que rare, il arrive parfois qu'une marque soit d'accord de sponsoriser une application. En échange, les espaces publicitaires de l'application sont entièrement occupés par la marque. C'est le cas notamment

de l'application du quotidien Libération qui s'est fait sponsoriser pendant un mois par une marque de parfum qui proposait gratuitement l'application payante.

Cette stratégie de communication a beaucoup plus de succès que la publicité faite sur des bannières (dans les applications), et les annonceurs commencent à s'intéresser de plus en plus à cette forme de campagne sponsorisée, si bien que des sites internet faisant l'intermédiaire entre les sponsors et les créateurs d'applications commencent à fleurir.

Il existe aussi la possibilité de développer une application qui sera uniquement utilisée en interne, et ne sera de ce fait pas disponible sur un Market. Ces applications sont destinées à être utilisés par les employés, ou parfois même par la clientèle de l'entreprise. C'est le cas par exemple de certains bar-restaurants qui affichent leur menu directement sur des tablettes distribuées au client lors de son arrivée.

Généralement ce genre d'application doit se connecter au serveur interne de l'entreprise afin de récupérer des données.

4.3 ***Les différentes versions d'Android***

Lorsque l'on développe des applications sur Android, on est confronté au problème de la fragmentation. Android a eu droit à plusieurs mises à jour depuis sa sortie dont des versions uniquement réservées aux tablettes. Aujourd'hui bien que les versions pour téléphones et pour tablettes soient les mêmes, diverses versions d'Android coexistent. Ces mises à jour ne sont pas disponibles pour tous les terminaux, mais seulement sur les dispositifs récents et haut de gamme. Il n'est d'ailleurs pas rare de voir sortir un mobile bas de gamme avec une ancienne version d'Android.

Une des particularités des différentes versions d'Android est qu'elles portent toutes un nom de dessert.

Figure 3
Distribution Actuelle des versions Android

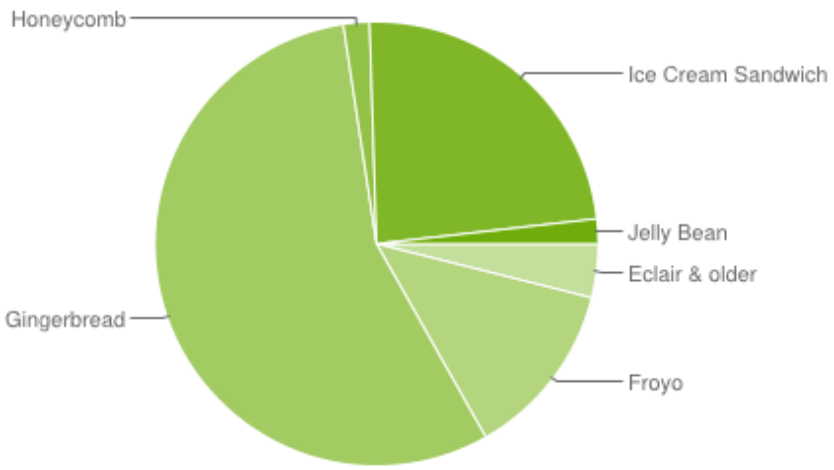


Tableau 1
Pourcentage de distribution actuelle des versions Android

Version	Codename	Distribution
1.5	Cupcake	0.1%
1.6	Donut	0.4%
2.1	Eclair	3.4%
2.2	Froyo	12.9%
2.3 – 2.3.7	Gingerbread	55.8%
3.1 – 3.2	Honeycomb	1.9%
4.03 - 4.04	Ice Cream Sandwich	23.7%
4.1	Jelly Bean	1.8%

Source developer.android.com (Octobre 2012)

Comme on peut le voir sur ce graphique la version la plus utilisée n'est pas la dernière, loin de là, mais c'est Gingerbread qui est présent sur plus de la moitié des dispositifs Android. Pendant le développement, il faut donc être particulièrement attentif à la compatibilité de l'application avec les anciennes versions d'Android.

La fragmentation ne s'arrête pas là, car plusieurs constructeurs d'appareils mobiles utilisent une surcouche logicielle, qui va légèrement modifier la version d'Android. Les plus connues sont « sense » de HTC ou « TouchWiz » de Samsung.

Hormis la version d'Android, il faut aussi tenir compte de la résolution ainsi de la taille de l'écran qui est différente sur presque chaque terminal. Afin de simplifier le design des interfaces utilisateurs Android divise la densité ainsi que la taille des écrans en un ensemble de quatre différentes tailles (small, normal, large, xlarge), ainsi qu'un ensemble de quatre différentes densités (ldpi, mdpi, hdpi, xhdpi).

Tableau 2
Pourcentage de résolution des écrans utilisés

	ldpi	mdpi	hdpi	xhdpi
small	1.7%		1.0%	
normal	0.4%	11%	50.1%	25.1%
large	0.1%	2.4%		3.6%
xlarge		4.6%		

Source developer.android.com (2012)

5. **Apache Maven**

Apache Maven, est un logiciel de gestion et d'automatisation de projet logiciel JAVA open source appartenant à la fondation Apache.

Maven est semblable à l'outil Apache Ant, car il possède une fonction pour faire du build, mais il est basé sur d'autres concepts et travaille d'une manière profondément différente. Il utilise un fichier XML qui s'appelle POM (Project Object Model) afin de décrire le projet logiciel en cours de construction, les dépendances[12] avec des modules externes, l'ordre de construction ainsi que les répertoires et les plug-ins nécessaires.

Maven télécharge dynamiquement les artefacts[13] sur un ou plusieurs dépôts logiciels connus (repository) ; il permet de spécifier quelle version de la librairie il faut utiliser.

En plus de ses fonctions de Build, Maven est capable de générer un site web, de générer des rapports ainsi que de faciliter la communication entre les membres d'une équipe de travail.

Maven étant très populaire, il est courant de voir que le code à rajouter dans le fichier POM.xml qui comprend les liens des dépendances, soit fourni par les distributeurs de librairies.

Les Maven dependencies se présentent sous la forme suivante :

```
<dependency>
    <groupId>le nom de l'organisation responsable du projet </groupId>
    <artifactId>le nom de la librairie (du JAR)</artifactId>
    <version>le nom et numéros de la version</version>
</dependency>
```

Un exemple de fichier POM.XML est disponible dans l'Annexe 2.

12 Une dépendance est une référence vers un artefact spécifique contenu dans un repository.

13 Un artefact est un élément spécifique issu de la construction du logiciel, tels que des librairies contenues dans des JARs, ou un fichier zip, ou un WAR

6. **JSON**

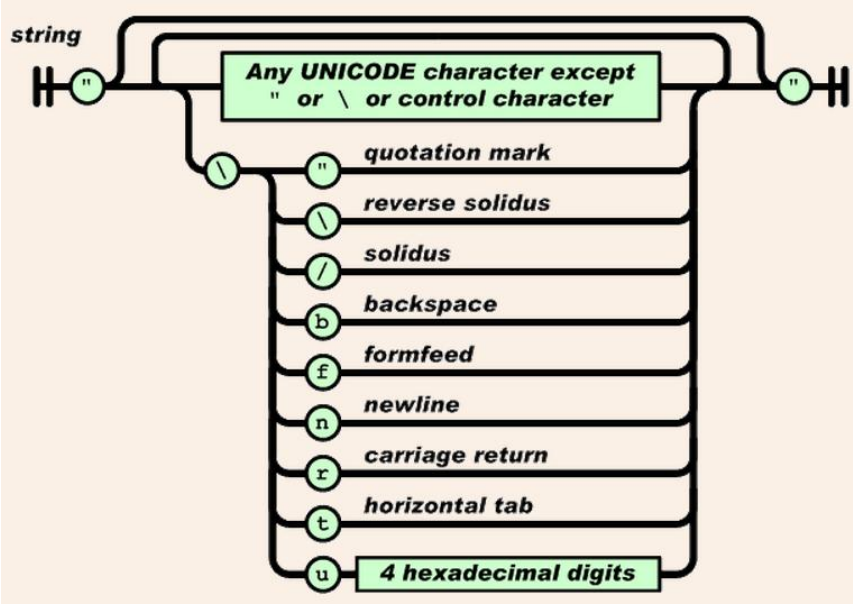
Json est un format léger d'échange de données textuel qui est facile à lire ou à écrire pour des humains ; il est indépendant de tout langage, mais les conventions qu'il utilise sont familières aux langages descendant du C, tel que le C++, le Java, JavaScript, Python, etc. Ce langage est souvent utilisé pour communiquer des informations entre différents langages de programmation.

JSON comprend deux éléments structurels :

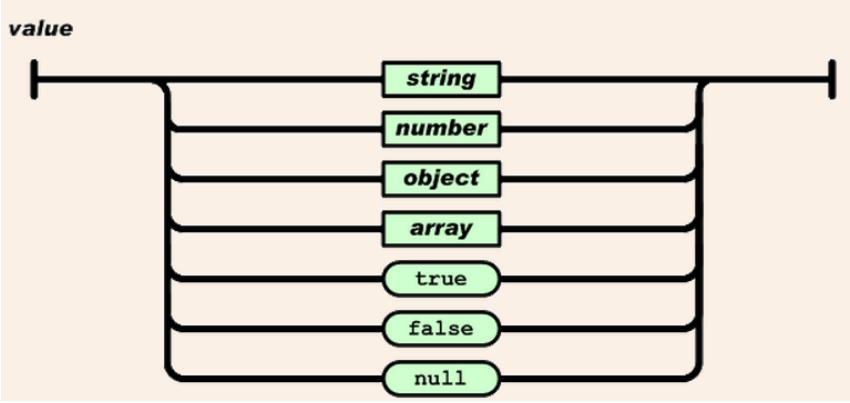
1. des ensembles de paires nom / valeur ;
2. une liste de valeurs ordonnées

Ces éléments structurels représentent différents types de données :

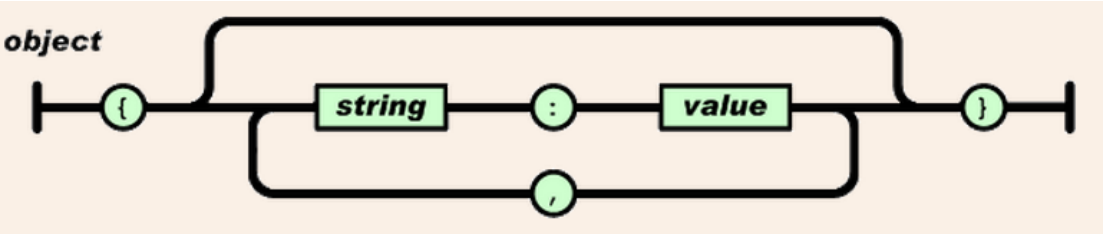
1. Une chaîne de caractères.



2. Une valeur

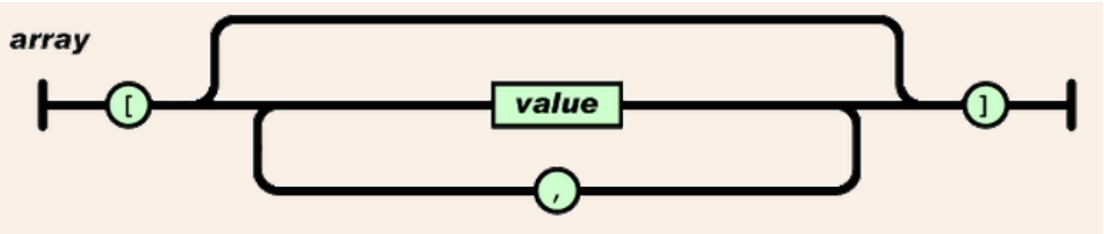


3. Un objet



Exemple : {"prenom":"Yuri","nom":"Allendes"}

4. Un tableau



Exemple d'array of integer : [1,2,3,4,5]

Source images : http://www.json.org/

Plusieurs librairies permettent de convertir des objets Java en JSON et vice-versa, Spring Android permet s'appuie sur deux librairies différentes à choix :

- Jackson qui est fournie sur le site http://jackson.codehaus.org/, et qui est certainement la librairie avec les meilleures performances parmi toutes les librairies JSON.
- Gson disponible sur le site http://code.google.com/p/google-gson/, qui est l'une des plus simple à utiliser et qui a l'avantage d'être moins volumineuse que la librairie Jackson.

7. **REST**

REST (Representational State Transfer) n'est pas un protocole, c'est un style d'architecture réseau pour web service décrit par Roy Fielding en 2000, qui repose sur le protocole HTTP afin de définir la sémantique de la communication client-serveur.

On accède aux ressources grâce à leur adresse URI en utilisant les messages du protocole HTTP soit :

- **GET** : pour récupérer une entité
- **POST** : pour la création d'une entité
- **PUT** : pour modifier ou créer une entité
- **DELETE** : pour l'effacement d'une entité
- **HEAD** : pour récupérer les informations sur l'entité

7.1 ***Les principes d'une architecture REST***

Une architecture RESTful doit adhérer à plusieurs principes :

- **Une architecture client-serveur :** séparation des contraintes
- **Les requêtes sont stateless (sans état)** ce qui veut dire que les requêtes sont indépendantes l'une de l'autre. Une nouvelle requête doit contenir tout ce qui est nécessaire à son traitement sans faire référence à une requête précédente.
- **Support des caches** : grâce aux requêtes GET, les informations peuvent être mises en cache.
- **Interface uniforme** : une interface est représentée de manière unique par une URI
- **Système hiérarchisé :** un composant ne doit se soucier que des composants avec qui il est en communication directe

7.2 ***Différence entre SOAP et REST***

Il existe plusieurs méthodes pour l'échange de données entre client et serveur. Parmi les plus utilisées on retrouve SOAP (Simple Object Access Protocol) et REST. Ces deux méthodes ne sont pas vraiment comparables, car ce sont deux approches complètement différentes. Comme nous l'avons vu, REST est un style d'architecture tandis que SOAP définit un protocole permettant des appels de procédures à distance.

Dans les illustrations suivantes nous pouvons voir qu'une des plus grandes différences entre les deux méthodes est le nom de la ressource qui se situe dans l'URI pour REST alors que pour SOAP les messages sont envoyés vers un point d'entrée unique.

Figure 4
Méthode de fonctionnement avec REST

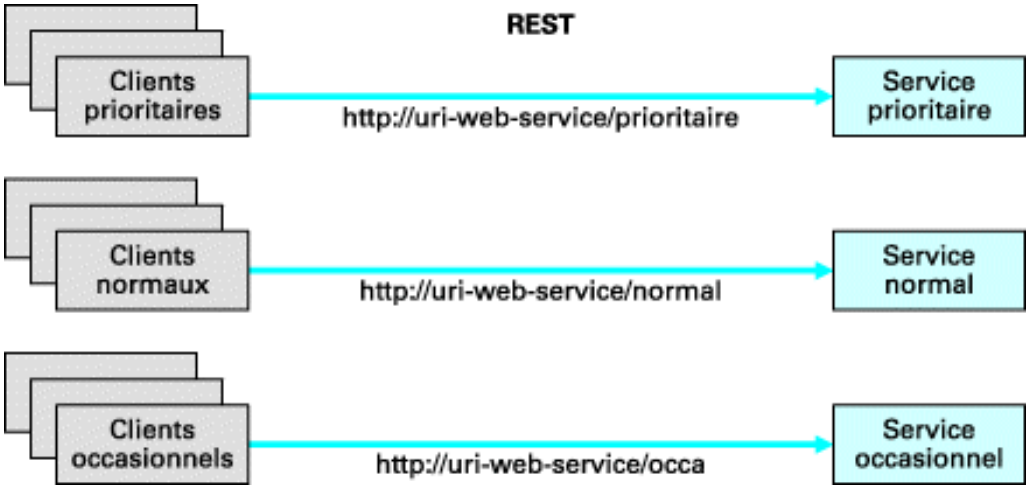
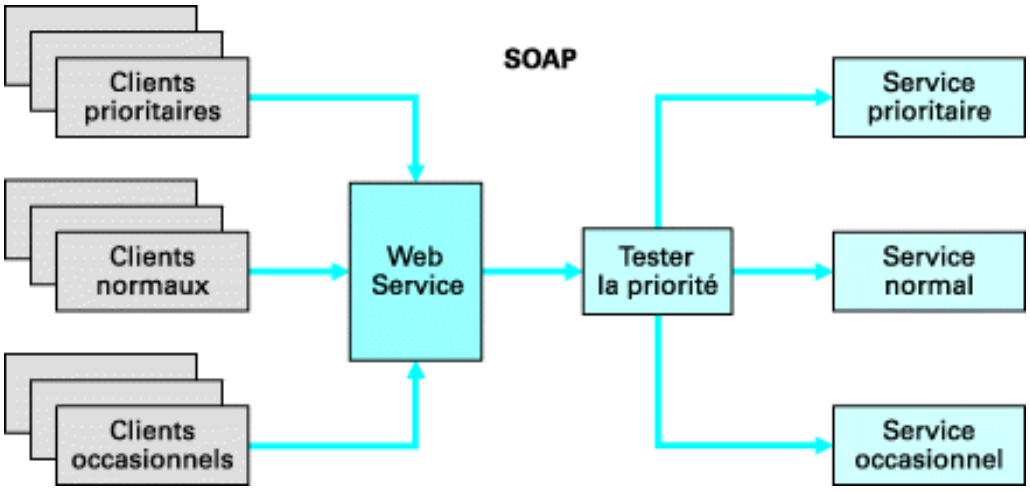


Figure 5
Méthode de fonctionnement avec SOAP



Source http://www.techniques-ingenieur.fr

Chacune des deux méthodes à ses avantages et inconvénients. REST sera choisi de préférence pour des services Web orientés ressources qui ne nécessitent que des opérations CRUD, alors que SOAP sera plutôt utilisé lorsqu'il est nécessaire de faire des opérations plus complexes que CRUD et lorsqu'un niveau de sécurité plus élevé est nécessaire (bien que la sécurité dans REST puisse être introduite par Spring Security par exemple).

Tableau 3
Avantage et inconvénient de REST

REST	
Avantages	**Désavantage**
Les résultats sont lisibles par un humain	Pas toujours adapté à des besoins transactionnels complexes
Cache réseau	Nécessité de mettre en place ses propres méthodes
Interface uniforme	
Performances	
Simplicité de mise en œuvre	
L'utilisation de multiples formats d'échange de données (JSON, XML, HTML)	

Tableau 4
Avantage et inconvénient de SOAP

SOAP	
Avantages	**Désavantage**
Les outils de développements sont disponibles dans pratiquement tous les langages de programmation	A cause du format XML qui est employé par SOAP les échanges peuvent être lourds
Bon niveau de sécurisation	
Le support (Microsoft/W3C)	

8. Caractéristique de Spring for Android

Le framework « Spring for Android » est fait pour être utilisé sur un projet Android uniquement. Pour l'utiliser il est nécessaire d'avoir un environnement de travail bien configuré, si ce n'est pas le cas veuillez-vous référer à l'Annexe 1.

Ce framework comprend trois librairies

- **Spring Android Rest Template :** qui permet de développer un client REST
- **Spring Android Auth :** qui permet d'utiliser des fonctionnalités de Spring Security
- **Spring Android Core** : qui contient des fonctionnalités communes aux deux autres modules

Ces trois librairies doivent être ajoutées manuellement dans le projet Android ou, dans le cas où l'on utilise MAVEN, les dépendances suivantes doivent être ajoutées dans le fichier POM.xml :

```
<dependency>
    <groupId>org.springframework.android</groupId>
    <artifactId>spring-android-rest-template</artifactId>
    <version>${spring-android-version}</version>
</dependency>
```

```
<dependency>
    <groupId>org.springframework.android</groupId>
    <artifactId>spring-android-auth</artifactId>
    <version>${spring-android-version}</version>
</dependency>
```

```
<dependency>
    <groupId>org.springframework.android</groupId>
    <artifactId>spring-android-core</artifactId>
    <version>${spring-android-version}</version>
</dependency>
```

Il n'est pas suffisant de n'ajouter que ces trois libraires car celles-ci sont dépendantes d'autres librairies tels que « spring social » ou encore « spring security ». Il n'est pas possible de donner une liste exhaustive des librairies qu'il faut rajouter car cela dépend de l'utilisation de l'application, mais vous pouvez trouver en annexe 2, un exemple de fichier POM.XML qui comprend la plupart des dépendances. Pour plus d'information veuillez-vous referez au site suivant :

https://github.com/SpringSource/spring-framework/wiki/SpringSource-repository-FAQ.

8.1 ***RestTemplate Module***

La principale fonctionnalité de Spring for Android est de proposer un client REST. Pour cela le framework utilise une version de RestTemplate (présent dans l'extension Spring Data) qui fonctionne dans un environnement Android.

8.1.1 Requête HTTP

La classe RestTemplate est donc le cœur de la première librairie, elle utilise les librairies internes d'Android afin de faire les requêtes http, mais il faut cependant lui préciser quel client HTTP utiliser, car Android en possèdent deux :

- Apache HTTP Client
- HTTP URLConnection

Google™ recommande d'utiliser le premier client pour les versions antérieures à Gingerbread (Android 2.3), et le deuxième pour les autres. Par défaut c'est le client approprié à la version Android employée qui est utilisé. On peut toutefois spécifier le client http qu'il faut utiliser à l'instanciation de la classe RestTemplate ou grâce à la méthode setRequestFactory.

Pour Apache HTTP Client:

```
HttpComponentsClientHttpRequestFactory
```

Pour HTTP URLConnection:

```
SimpleClientHttpRequestFactory
```

8.1.1.1 Exemple

A l'instanciation de la class RestTemplate :

```
ClientHttpRequestFactory client =
       new HttpComponentsClientHttpRequestFactory();
RestTemplate restTemplate = new RestTemplate(client);
```

Avec la méthode setRestFactory :

```
ClientHttpRequestFactory client =
       new HttpComponentsClientHttpRequestFactory();
RestTemplate restTemplate = new RestTemplate();
restTemplate.setRequestFactory(client);
```

8.1.2 Convertisseurs

Lors d'une requête, les objets transmis et récupérés sont convertis par les instances de HttpMessageConverter. Voici la liste exhaustive des convertisseurs disponibles, cependant il est également possible d'écrire son propre convertisseur

- **ByteArrayHttpMessageConverter :** peut lire et écrire des tableaux d'octets
- **FormHttpMessageConverter :** peut lire et écrire des données de formulaire.
- **XmlAwareFormHttpMessageConverter :** est une extension de FormHttpMessageConverter ajoutant le support XML
- **ResourceHttpMessageConverter :** peut lire et écrire des ressources
- **SourceHttpMessageConverter :** peut lire et écrire un transformeur (java.xml.tranform.Source)
- **StringHttpMessageConverter :** peut lire et écrire des Strings
- **SimpleXmlHttpMessageConverter :** peut lire et écrire des données XML en utilisant le framework « Simple Framework »
- **MappingJacksonHttpMessageConverter :** peut lire et écrire en JSON en utilisant la librairie « Jackson »
- **GsonHttpMessageConverter :** peut lire et écrire en JSON à en utilisant la librairie « Gson »
- **SyndFeedHttpMessageConverter :** peut lire et écrire des flux RSS et ATOM en utilisant la librairie « android rome feed reader »
- **RssChannelHttpMessageConverter :** peut lire et écrire des flux RSS
- **AtomFeedHttpMessageConverter :** peut lire et écrire des flux Atom

8.1.2.1 Exemple

Voici un exemple utilisant un convertisseur de XML. Pour que cet exemple fonctionne il est nécessaire de rajouter la librairie du framework « simple framework » au projet.

```
// Création d'une instance de RestTemplate
RestTemplate restTemplate = new RestTemplate();

// Ajout du Convertisseur
restTemplate.getMessageConverters().add(
       new SimpleXmlHttpMessageConverter());
```

Tous les convertisseurs utilisant une autre librairie nécessite l'ajout de celle-ci dans le projet.

8.1.3 Requête HTTP

Après avoir défini notre convertisseur on peut faire une requête HTTP. Tous les messages du protocole HTTP peuvent être invoqués :

HTTP	RestTemplate
GET	getForObject(java.lang.String, java.lang.Class, java.lang.Object...)
	getForEntity(java.lang.String, java.lang.Class, java.lang.Object...)
POST	postForLocation(java.lang.String, java.lang.Object, java.lang.Object...)
	postForObject(java.lang.String, java.lang.Object, java.lang.Class, java.lang.Object...)
PUT	put(java.lang.String, java.lang.Object, java.lang.Object...)
DELETE	delete(java.lang.String, java.lang.Object...)
HEAD	headForHeaders(java.lang.String, java.lang.Object...)

8.1.3.1 Exemple:

Voici un exemple de requête GET qui récupère un String.

```
// Création d'une instance de RestTemplate
RestTemplate restTemplate = new RestTemplate();
// Ajout du Convertisseur
restTemplate.getMessageConverters().add(
        new StringHttpMessageConverter());
// Récupération du String
String résultat = restTemplate.getForObject(url, String.class);
```

8.1.4 Compression Gzip

Il est également possible d'utiliser la compression Gzip, ceci permet de réduire considérablement la taille des données envoyées ou reçues. Pour utiliser cette fonctionnalité il faut que le serveur avec lequel on communique prenne en charge ce type de compression. On spécifie alors le type du codage (ContentCodingType) dans l'en-tête (header) de la requête afin que la réponse du serveur se fasse en utilisant la compression Gzip. Si la compression Gzip est disponible sur le serveur alors la réponse renvoyée pas le serveur sera compressée. Le RestTemplate va vérifier automatiquement dans l'en-tête (header) de la réponse afin de savoir si la réponse est

compressée ou non. Si c'est le cas alors un GZIPINputStream sera utilisé pour décompresser la réponse.

```
HttpHeaders requestHeaders = new HttpHeaders();
requestHeaders.setAcceptEncoding(ContentCodingType.GZIP);
HttpEntity<?> requestEntity = new HttpEntity<Object>(requestHeaders);
```

8.2 ***Auth Module***

De nos jours, la majorité des applications mobiles se connectent à des services Web (SaaS) de manière à accéder à des données de toutes sortes. Dans la plupart des cas, afin d'accéder à ces données, il est nécessaire de s'authentifier (nom d'utilisateur, adresse email, mots de passe), L'objectif du module « Android auth » est de simplifier la connexion et donc l'authentification avec les services web. Ces informations peuvent ensuite être stockées sur le téléphone. Le module fournit aussi un moyen d'encrypter des données telles que les mots de passes.

8.2.1 Authentification

La première étape est de créer une connexionFactory correspondant au SaaS utilisé, puis la seconde est d'établir la connexion OAuth.

8.2.1.1 Exemple Facebook

L'appID et l'appSecret sont fournis par Facebook lorsqu'on inscrit une application à l'adresse suivante : https://developers.facebook.com/

```
//Creation Factory
FacebookConnectionFactory facebookConnectionFactory;
facebookConnectionFactory =
       new FacebookConnectionFactory(appId, appSecret);

String redirectUri =
"https://www.facebook.com/connect/login_success.html";
String scope =
"publish_stream,offline_access,read_stream,user_about_me";

//Paramètres Oauth
OAuth2Parameters parameters = new OAuth2Parameters();
parameters.setRedirectUri(redirectUri);
parameters.setScope(scope);
parameters.add("display","touch");

//Connexion Oauth
OAuth2Operations oauth = connectionFactory.getOAuthOperations();
String authorizeUrl =
       oauth.buildAuthorizeUrl(GrantType.IMPLICIT_GRANT, parameters);


AccessGrant accessGrant = new AccessGrant(accessToken);
Connection<Facebook> connection =
       facebookConnectionFactory.createConnection(accessGrant);
```

8.2.2 Stockage de la connexion

Les informations concernant la connexion peuvent être stockées sur la base de données interne SQLite grâce à la classe « SQLiteConnectionRepository ». Ce repository est conçu pour stocker les informations de connexion aux multiples fournisseurs de services Web d'un utilisateur et il est même possible de stocker plusieurs comptes, au même fournisseur de service, qu'un utilisateur pourrait avoir.

Cette base de données pourrait, par exemple, être utilisée dans une application Agenda qui accéderait aux informations telles que les rendez-vous stockés dans divers autres agendas (agenda Google™, agenda Hotmail®, etc...)

Il existe aussi une classe « SQLiteUsersConnectionRepository » qui permet de stocker de multiples comptes utilisateurs ; cette classe n'as pas beaucoup d'intérêt pour l'instant car il est rare que des personnes se partagent un téléphone. Par contre, avec l'arrivée prochaine d'Android 4.2 et sa nouvelle fonctionnalité permettant le multi-compte, cette classe peut prendre toute son importance.

9. Exemple concret d'application

9.1 ***Introduction***

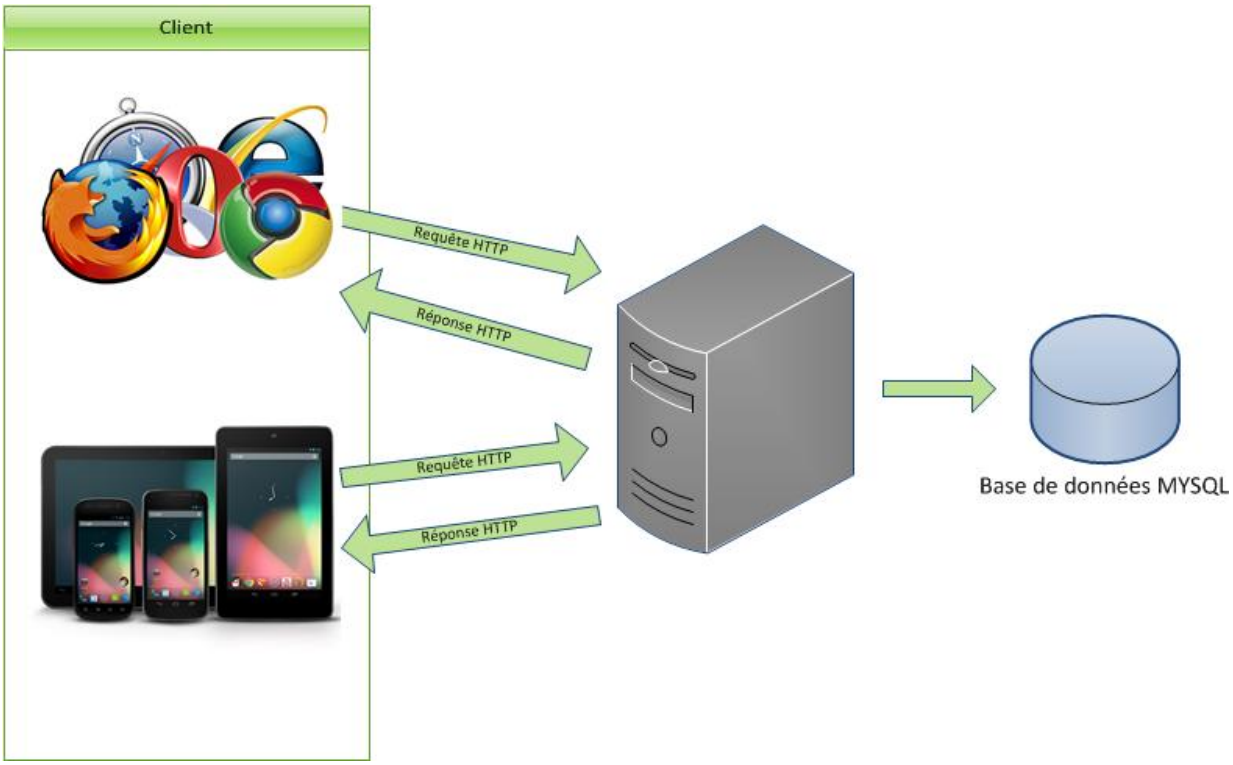
Afin de mettre en œuvre le framework Spring for Android d'une façon concrète, j'ai créé une application Android (Afin d'importer l'application sur Eclipse, veuillez-vous referez à l'annexe 1)

Cette application utilise le module RestTemplate pour faire les requêtes au serveur d'application, avec lequel il communique en JSON.

Il est également possible de communiquer avec le serveur d'application depuis un navigateur internet.

Le serveur d'application est relié à une base de données de façon à stocker et à accéder aux données.

Figure 6
Vue d'ensemble du projet

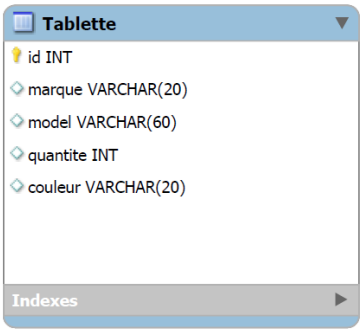


9.2 ***Contexte***

Notre client gère un stock de différents modèles de tablettes. Il désire, depuis son téléphone Android, pouvoir consulter la liste des tablettes disponibles dans son stock à tout moment, ainsi qu'ajouter un nouveau modèle de tablette à la liste.

9.3 ***Base de Données***

La base de données ne contient qu'une table « Tablette » et l'identifiant de la table « id » s'auto incrémente. Le SGBD[14] utilisé est MySQL.

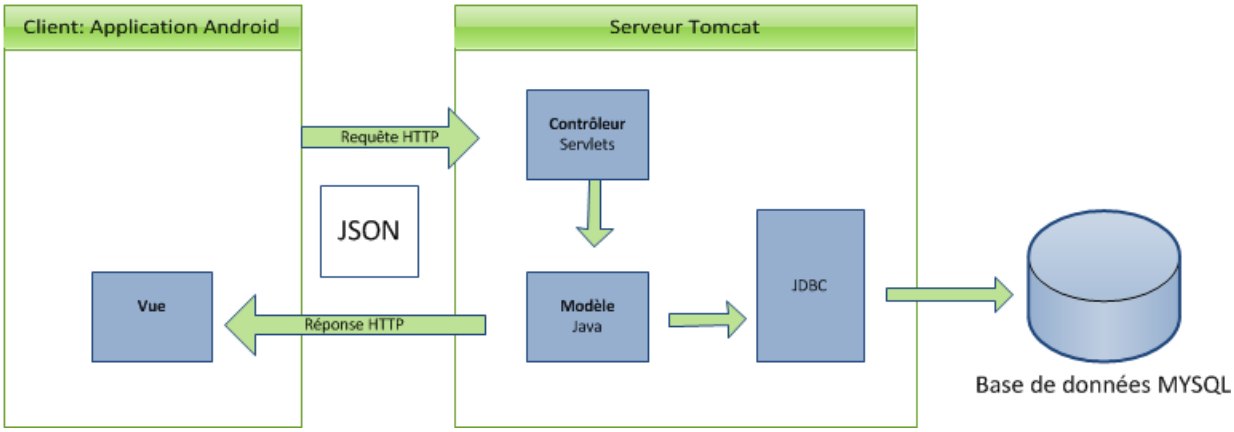


9.4 ***Mise en œuvre***

Avant de commencer à programmer l'application Android il a fallu mettre en place une application Java EE sur un serveur d'application. Cette application va devoir traiter les requêtes HTTP faites par le client.

9.4.1 Client Android

Figure 7
Vue d'ensemble Client Android



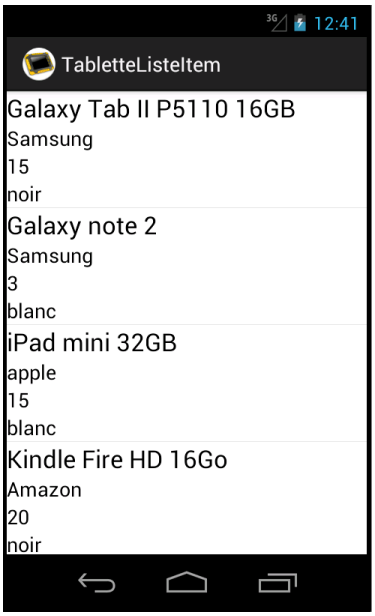
Lorsque le client envoie une requête « HTTP GET » le serveur va chercher la liste des tablettes présentes sur la base de données et les renvoyer en format JSON.

L'application va ensuite directement transformer les données JSON en classe java grâce au convertisseur Gson ajouté dans le RestTemplate :

14 Système de gestion de base de données

```
restTemplate.getMessageConverters().add(
        new GsonHttpMessageConverter());
TabletteList tablettesListe =
        restTemplate.getForObject(url, TabletteList.class);
```

L'application va se charger ensuite d'afficher les données dans une liste.



Lorsque le client envoie une requête « HTTP POST » qui contient un objet « tablette », en format JSON ; le serveur va récupérer cet objet, le transformer en objet Java et l'ajouter à la base de données, puis renvoyer une réponse pour dire que le traitement s'est bien passé.

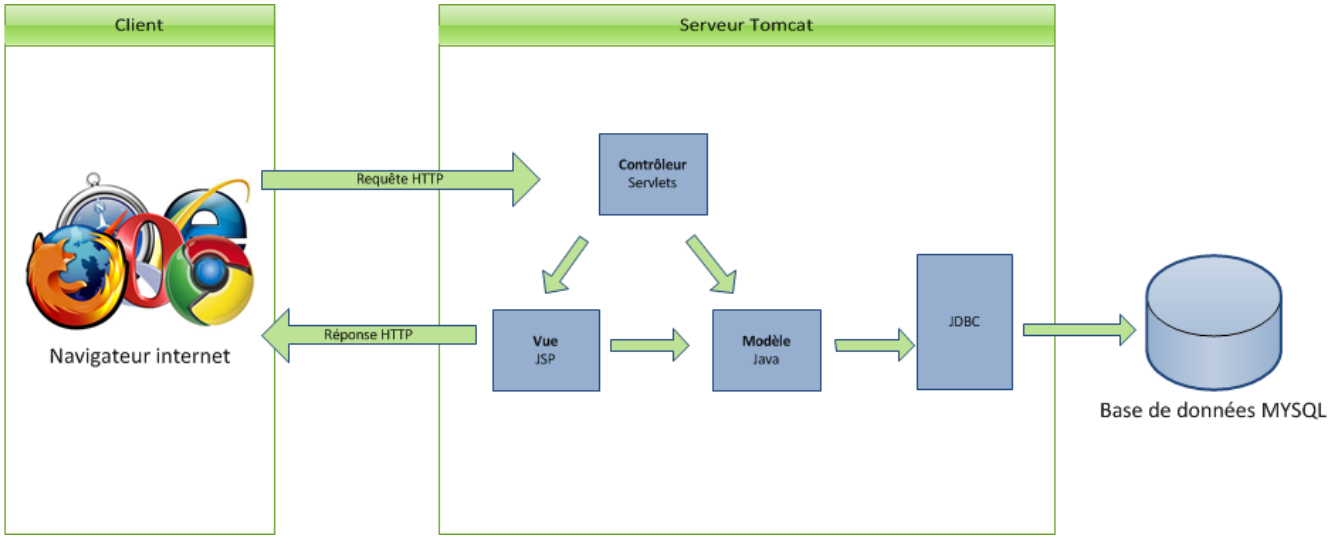
Il est nécessaire d'ajouter deux convertisseurs au RestTemplate, un pour la conversion de l'objet `Tablette` qui va être envoyé au serveur et l'autre pour la réception du message de succès (si tout s'est bien passé).

```
//Ajout des convertisseurs
restTemplate.getMessageConverters().add(
        new GsonHttpMessageConverter());
restTemplate.getMessageConverters().add(
        new StringHttpMessageConverter());
// Requête POST envoie de l'objet et récupération du message
String response = restTemplate.postForObject(url, tab, String.class);
```

9.4.2 Client Navigateur

Dans le but de pouvoir tester que l'application Java EE fonctionne bien, des JSP [15] ont été mis en place afin de créer dynamiquement des pages web. Ceci permet de faire fonctionner l'application depuis un navigateur internet. Dans ce cas-là le client est le navigateur. Le servlet va intercepter la requête HTTP, puis appeler le modèle pour qu'il fasse les traitements nécessaires et ensuite ordonner à la Vue (JSP) d'afficher le résultat au navigateur internet.

Figure 8
Vue d'ensemble Client Navigateur Internet



[15] JavaServer Page

Conclusion

Ce mémoire aura permis d'analyser les différentes fonctionnalités proposées par le framework « Spring for Android » et d'en comprendre le fonctionnement ainsi que l'utilité. L'application Android développée, bien qu'étant simple, est un exemple concret de ce qu'il est possible de réaliser avec l'aide du framework et elle aura permis de mettre en pratique quelques-unes des fonctionnalités évoquées.

L'utilité d'un framework n'est plus à démontrer et comme nous l'avons vu, Android étant en train de monter en puissance, il était normal de voir apparaitre des frameworks compatibles avec l'OS. Ces frameworks étant récents, ils ne disposent pas d'une très grande communauté et ne possèdent donc pas beaucoup de documentation, ou d'exemples concrets.

Avec l'arrivée de « Spring for Android », les choses sont amenées à changer, car la communauté de Spring est très active, le framework étant déjà bien implanté dans le milieu. Il est l'un des seuls framework actuels à proposer un client REST pour Android et indéniablement sa facilité d'utilisation nous fait gagner un temps précieux en développement.

Mais le framework n'est pas exempt de défauts. En effet la taille des librairies est un peu volumineuse, 311Ko pour les trois librairies principales auquel il faut ajouter le poids de toutes les dépendances que celles-ci peuvent avoir avec d'autres librairies (exemple : spring security, spring social, Jackson, Gson, etc.). Le framework a beaucoup de dépendance avec d'autres librairies et il en résulte parfois des incompatibilités, c'est pourquoi l'utilisation d'un outil tel que Maven est presque obligatoire. L'installation de Maven sur Android n'est pas des plus faciles et pour ceux qui n'ont jamais utilisé ce gestionnaire de projet, un apprentissage supplémentaire est nécessaire, ce qui peut décourager l'utilisateur novice voulant utiliser « Spring for Android »

Néanmoins, il est selon moi, le meilleur et le plus complet des frameworks disponibles sur Android et même si la taille des librairies peut être imposante, elle en vaut la peine, le framework nous facilitant grandement le développement des applications.

Ainsi que nous avons pu le constater « Spring for Android » est plutôt une adaptation pour Android de différentes fonctionnalités présentes dans d'autres extensions de Spring. Il est donc probable que dans l'avenir d'autres fonctionnées soient adaptées à ce framework plein d'avenir.

Il est aussi envisageable qu'à l'avenir de plus en plus de frameworks Java bien connus et disposant eux aussi d'une grande communauté, adaptent leur framework pour qu'ils fonctionnent sur Android.

N'ayant jamais utilisé de framework auparavant, ce travail m'aura permis de mettre un pied dans l'univers des frameworks. Il m'aura aussi été permis d'apprendre à développer une application web Java EE, et ainsi de comprendre des concepts tels que la structure d'une application Java EE, ou encore ce qu'est un servlet. J'ai aussi dû utiliser plusieurs langages avec lesquels je n'étais pas familier, comme le JSP ou le JSTL ainsi que le JSON.

J'ai aussi dû configurer et utiliser l'outil MAVEN, qui me sera certainement d'une grande utilité par la suite, tant il semble être populaire dans le monde des frameworks.

Enfin, il m'a aussi été possible de mettre en pratique un pattern d'architecture (MVC) vu dans le cours « Architecture des systèmes d'information » et d'améliorer mes connaissances sur la programmation Android pratiqué dans le cours de Programmation.

Bibliographie

CHING Maria Odea, PORTER Brett, *Apache Maven 2 Effective Implementation: Build and manage applications with Maven, Continuum, and Archiva*, Birmingham, Packt Publishing, 2009.

HARROP Rob, HO Clarence, Pro Spring 3, New York, Edition Apress, 2012.

MAK Gary, Spring par l'exemple, Paris, Editions Pearson Education France, 2009.

MEAUDRE Adrien, Java et Spring: Concevoir, construire et développer une application Java/J2EE avec Spring, St Herblain : Editions ENI, 2010.

ROD Johnson, Expert One-onOne J2EE Design and Development, Indianapolis, Indiana, Edition Wiley Publishing, 2003.

Webographie

ANDROID BINDING FRAMEWORK [en ligne] http://code.google.com/p/android-binding/ (consulté le 15.11.2012)

ANDROID OFFICIAL BLOG -officialandroid.blogspot.ch [en ligne] http://officialandroid.blogspot.ch/2012/09/google-play-hits-25-billion-downloads.html (consulté le 15.11.2012)

API ANDROID - developer.android.com [en ligne] http://developer.android.com/reference (consulté le 15.11.2012)

API SPRING - static.springsource.org [en ligne] http://static.springsource.org/spring-android/docs/1.0.x/api/ (consulté le 14.11.2012)

BENKIRANE, Amine - Benkirane.blog.parisjob.com [en ligne] http://benkirane.blog.parisjob.com/index.php/post/2012/05/Architecture-logicielle-:-SOAP-vs.-REST-Web-services (consulté le 14.11.2012)

BLOG DEVELOPPER. SPRING SOCIAL - blog-dev.net [en ligne] http://blog-dev.net/tag/spring-social (consulté le 15.11.2012)

COMPRENDRE JSON - tomsyweb.com [en ligne] http://www.tomsyweb.com/component/content/article/44-javascript/80-comprendre-json (consulté le 15.11.2012)

DEVELOPPEZ SECTION JAVA. TUTO MAVEN - Developpez.com [en ligne] http://java.developpez.com/faq/maven/?page=terminologie (consulté le 14.11.2012)

DOUDOUX, Jean-Michel - Jmdoudoux.fr [en ligne] http://www.jmdoudoux.fr/java/dej/chap-frameworks.htm (consulté le 14.11.2012)

FLURRY - blog.flurry.com [en ligne] http://blog.flurry.com/bid/83604/For-Generating-App-Revenue-Amazon-Shows-Google-How-to-Play (consulté le 15.11.2012)

GON USER GUIDE - sites.google.com [en ligne] https://sites.google.com/site/gson/gson-user-guide (consulté le 15.11.2012)

HILLERT, Gunnar - hillert.blogspot.ch [en ligne] http://hillert.blogspot.ch/2011/01/rest-with-spring-contentnegotiatingview.html (consulté le 15.11.2012)

HOW DO I CONVERT COLLECTIONS INTO JSON ? - kodejava.org [en ligne] http://www.kodejava.org/examples/588.html (consulté le 15.11.2012)

JAVA JDK DOWNLOAD- oracle.com [en ligne] http://www.oracle.com/technetwork/java/javase/downloads/index.html (consulté le 15.11.2012)

JSON – json.org [en ligne] http://www.json.org/ (consulté le 15.11.2012)

JSON OBJECT PASSING - stackoverflow.com [en ligne] http://stackoverflow.com/questions/3379586/json-object-passing-help (consulté le 14.11.2012)

LE SPONSORING - journaldunet.com [en ligne] http://www.journaldunet.com/ebusiness/internet-mobile/rentabiliser-application-gratuite/le-sponsoring.shtml (consulté le 15.11.2012)

LE SPONSORING SUR APPLICATION MOBILE - connecting-sponsors.fr [en ligne] http://www.connecting-sponsors.fr/actualites-le_sponsoring_sur_application_mobile (consulté le 15.11.2012)

LEONARDI FRAMEWORK [en ligne] http://www.leonardi-free.org (consulté le 15.11.2012)

MAVEN - maven.apache.org [en ligne] http://maven.apache.org (consulté le 15.11.2012)

MAVEN ANDROID PLUGIN - code.google.com [en ligne] http://code.google.com/p/maven-android-plugin (consulté le 15.11.2012)

MAVEN ANDROID SDK DEPLOYER - github.com [en ligne] https://github.com/mosabua/maven-android-sdk-deployer (consulté le 15.11.2012)

PLAY FRAMEWORK [en ligne] http://www.playframework.org (consulté le 15.11.2012)

RESTLET - Restlet.org [en ligne] http://www.restlet.org (consulté le 14.11.2012)

ROBOGUICE FRAMEWORK [en ligne] http://code.google.com/p/roboguice (consulté le 15.11.2012)

SHRIVANSH, Himanshu - shrivanshh.blogspot.ch [en ligne] http://shrivanshh.blogspot.ch/2010/04/spring-core-modules-ioc-aop.html (consulté le 14.11.2012)

SOAP VS REST - clever-age.com [en ligne] http://www.clever-age.com/veille/blog/soap-vs-rest-choisir-la-bonne-architecture-web-services.html (consulté le 15.11.2012)

SPRING - springsource.org [en ligne] http://www.springsource.org (consulté le 15.11.2012)

SPRINGSOURCE - static.springsource.org [en ligne] http://static.springsource.org/spring/docs/2.0.x/reference/introduction.html (consulté le 14.11.2012)

STRUTS 2 FRAMEWORK [en ligne] http://struts.apache.org/2.x/ (consulté le 15.11.2012)

TAPESTRY FRAMEWORK [en ligne] http://tapestry.apache.org (consulté le 15.11.2012)

THIBAULT, Vincent - commentrentabilisersonsiteweb.com [en ligne] http://www.commentrentabilisersonsiteweb.com/6-moyens-rentabiliser-application-mobile-iphone-android-generer-revenus (consulté le 15.11.2012)

TOMCAT MYSQL CONNECTION - mulesoft.com [en ligne] http://www.mulesoft.com/tomcat-mysql (consulté le 14.11.2012)

VAADIN FRAMEWORK [en ligne] https://vaadin.com/home (consulté le 15.11.2012)

WIKIPEDIA - Accueil [en ligne] http:/wikipedia.org (consulté le 15.11.2012)

WILSON, Jesse - Android-developers.blogspot.ch [en ligne] http://android-developers.blogspot.ch/2011/09/androids-http-clients.html (consulté le 14.11.2012)

Annexe 1
Configuration de l'environnement de travail

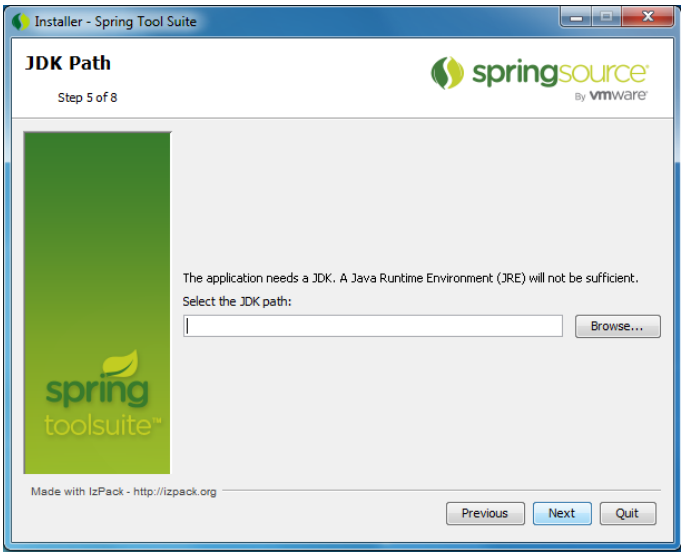
Avant d'utiliser le framework Spring for Android il faut configurer l'environnement de travail. Pour cela il faut tout d'abord être sûr d'avoir installé le JDK (Java Development Kit), si ce n'est pas le cas rendez-vous sur la page d'oracle et téléchargez :

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Après avoir installé le JDK, il va falloir choisir son IDE[16], Celui qui se prête le mieux à l'utilisation du framework est Eclipse et plus précisément Eclipse Spring TOOL SUITE (v. 3.1.0) qui est basé sur Eclipse JUNO 4.2.1. Si le choix s'est porté sur cet IDE plutôt qu'un autre c'est pour deux raisons : la première est qu'Eclipse est plus pratique pour le développement Android, et la seconde est que plusieurs modules relatifs à Spring sont déjà présents dans cette version.

L'adresse de téléchargement : http://www.springsource.org/spring-tool-suite-download

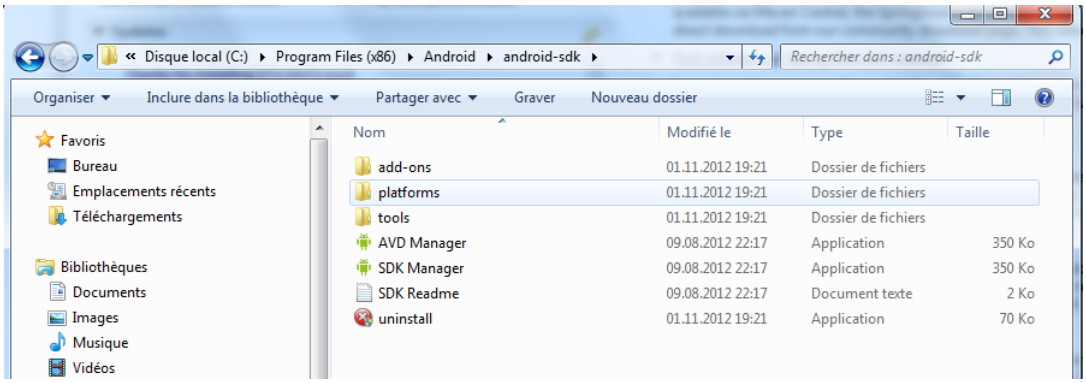
Pendant l'installation, il va falloir désigner le JDK précédemment installé (par défaut si vous êtes sous Windows il devrait se trouver sous C:\Program Files\Java\jdk1.7.0_09).



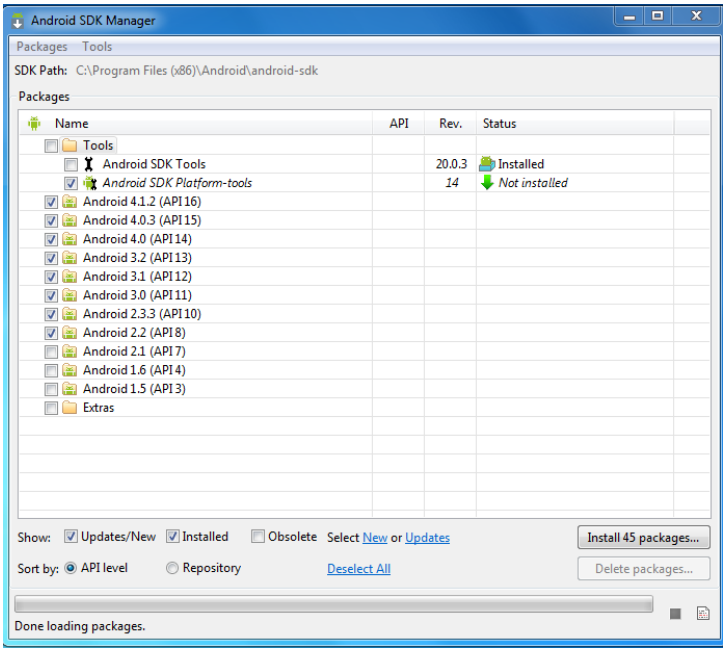
Notre IDE étant installé il va maintenant falloir télécharger et installer l'Android SDK depuis l'adresse suivante : http://developer.android.com/sdk/index.html.

16 L'IDE (Integrated Development Environment) est une interface qui permet de développer, compiler et exécuter un programme dans un langage donné.
Source, dico du net : http://www.dicodunet.com/definitions/developpement/ide.htm

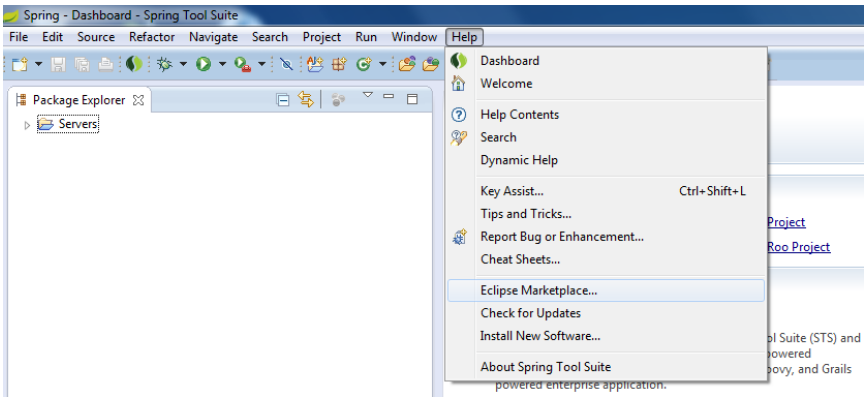
Une fois l'installation terminée, il vous est proposé de lancer le SDK Manager. Acceptez, ou allez l'ouvrir directement à l'endroit où vous avez installé l'Android SDK.



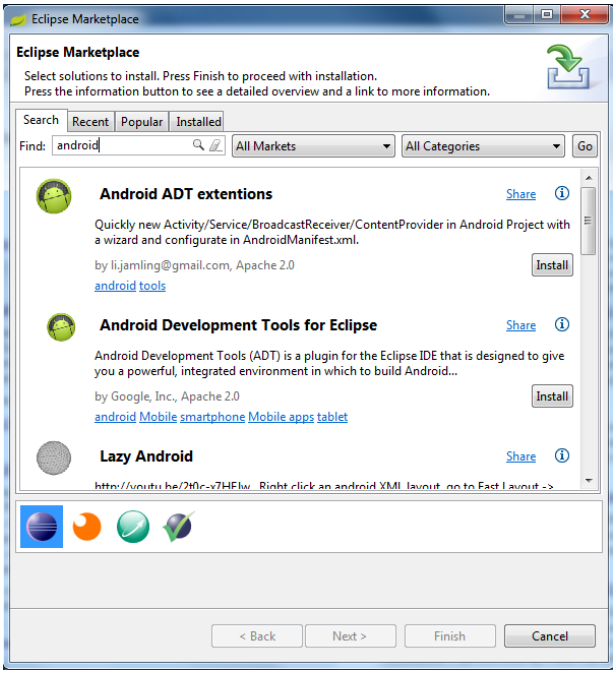
Sélectionnez l'Android SDK Plateform-tools ainsi que la ou les versions d'Android sur lesquelles vous voulez développer une installation (si possible toutes les versions depuis Android 2.2) et appuyer sur le bouton Install xx packages.



Ouvrez maintenant Spring Tool suite (STS) et choisissez un dossier pour votre espace de travail (Workspace). Une fois l'IDE lancé, allez sous l'onglet Help et cliquez sous Eclipse Marketplace.



Puis faites une recherche avec le mot clef « android », le plugin « Android Development tools for Eclipse » devrait apparaitre dans la liste. Installez-le, ce qui aura pour effet de redémarrer l'IDE. L'installation du plugin n'est pas obligatoire mais elle simplifie grandement la tache de l'intégration du SDK Android à l'IDE.

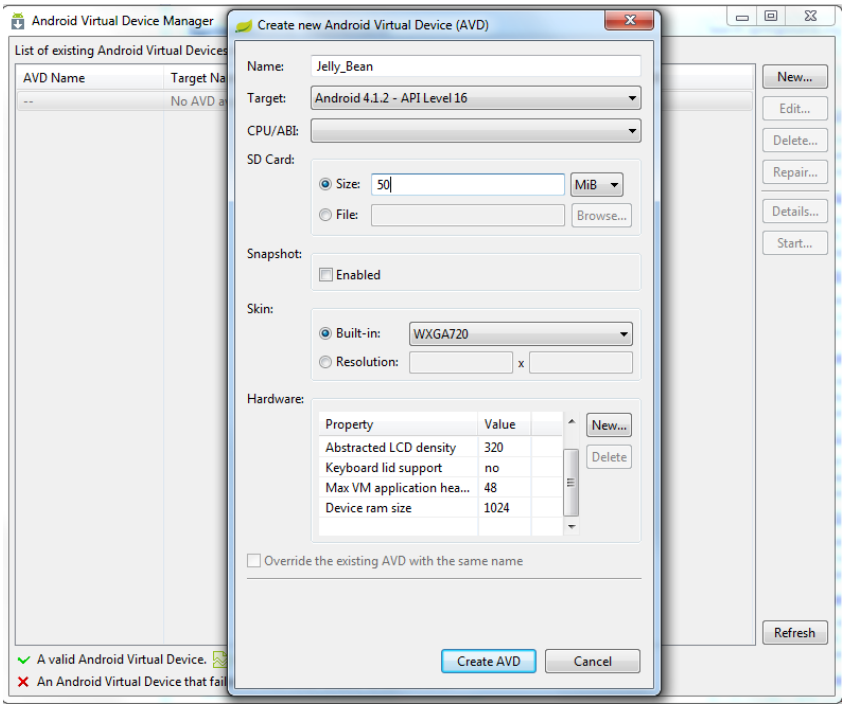


L'étape suivante va être de créer des machines virtuelles (Android Virtual Device AVD). Pour cela il faut se rendre sous Windows/ AVD Manager

Cliquez sur le bouton New et rentrez un nom pour votre AVD, sélectionnez la version d'Android désirée ainsi que les options de la machine virtuelle telles que la capacité de stockage et la résolution de l'écran.

Voici à quoi correspondent les résolutions disponibles :

- QVGA (240x320, low density, small screen)
- WQVGA400 (240x400, low density, normal screen)
- WQVGA432 (240x432, low density, normal screen)
- HVGA (320x480, medium density, normal screen)
- WVGA800 (480x800, high density, normal screen)
- WVGA854 (480x854 high density, normal screen)
- WXGA720 (1280x720, extra-high density, normal screen)
- WSVGA (1024x600, medium density, large screen)
- WXGA800-7in (1280x800, high density, large screen) new
- WXGA800 (1280x800, medium density, xlarge screen)



Notre environnement de travail est maintenant configuré, mais avant de commencer il nous faut encore mettre en place Maven.

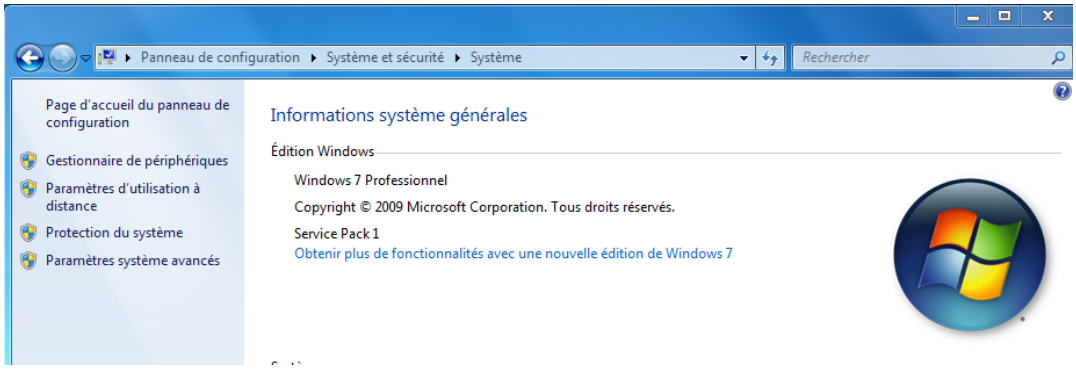
L'emploi de Maven n'est pas obligatoire lors de l'utilisation du framework Spring, mais il est fortement recommandé, car le framework utilise de nombreuses librairies qui dépendent parfois d'autres librairies (ne faisant pas forcément partie de la galaxie Spring) et il est bien plus simple de gérer les dépendances en utilisant Maven.

Maven n'a pas été conçu pour fonctionner avec Android, mais heureusement il existe un plugin « Maven Android Plugin » qui permet son utilisation. Pour utiliser le plugin il va falloir ajouter deux variables d'environnement supplémentaire et modifier le path ainsi que les dépendances du plugin dans notre fichier POM.xml.

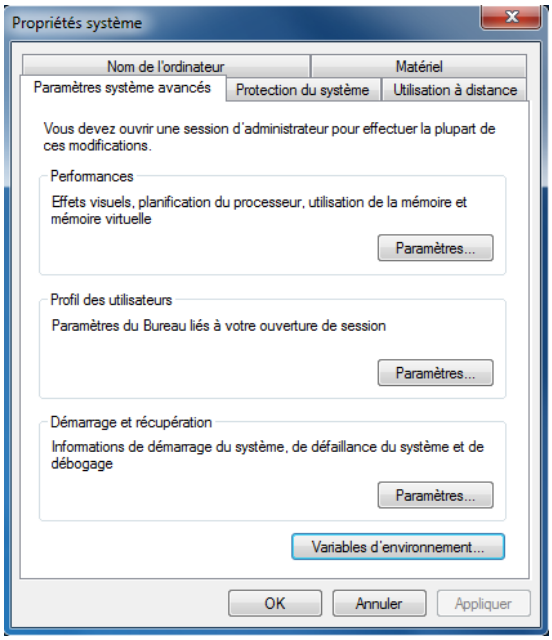
Tout d'abord, télécharger la dernière version de Maven sur le site d'apache Maven (version binary) :

http://maven.apache.org/download.html

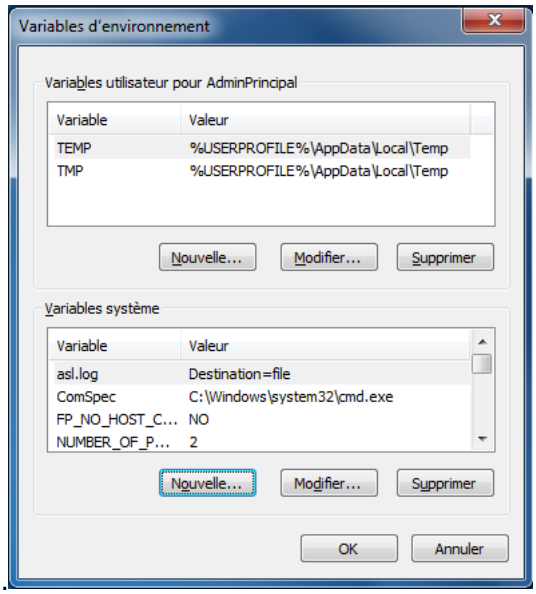
Si vous êtes sous Windows, décompressez le fichier et copiez-le dans vos programmes (C:\Program Files\apache-maven-3.0.4). Il va maintenant falloir déclarer deux variables d'environnement nécessaires au fonctionnement de Maven dans le système d'environnement de variable, ainsi que les deux autres variables nécessaires à l'utilisation du plugin « Maven Android Plugin ». Pour cela rendez-vous dans les propriétés système (Panneau de configuration\Système et sécurité\Système et cliquez sur « Paramètres système avancés »).



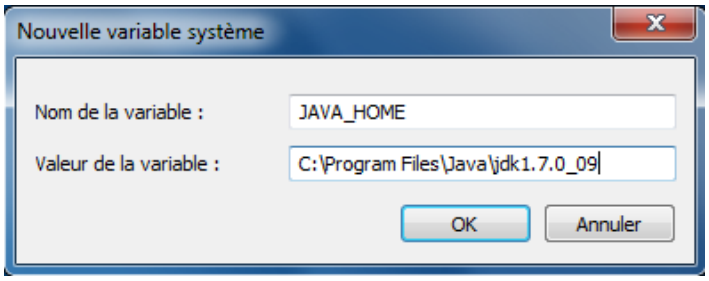
Puis cliquez sur le bouton « Variables d'environnement » qui se trouve tout en bas sous l'onglet Paramètres système avancés.

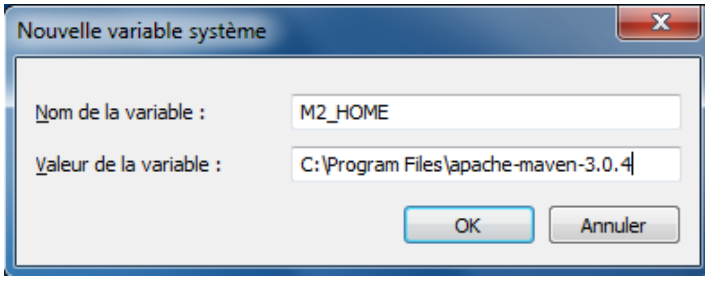


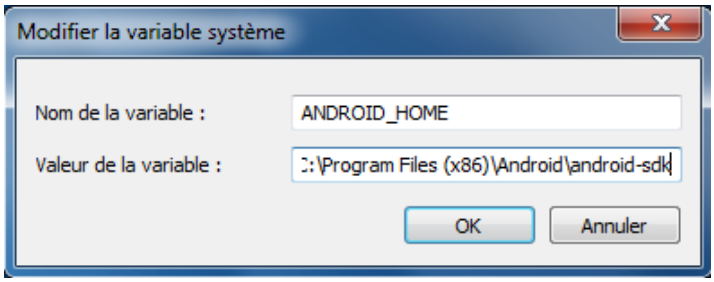
Il va maintenant falloir créer les variables d'environnement que l'on ajoutera par la suite au path. Afin de les créer appuyer sur le bouton « Nouvelle… » (le bouton entouré en bleu sur l'image)



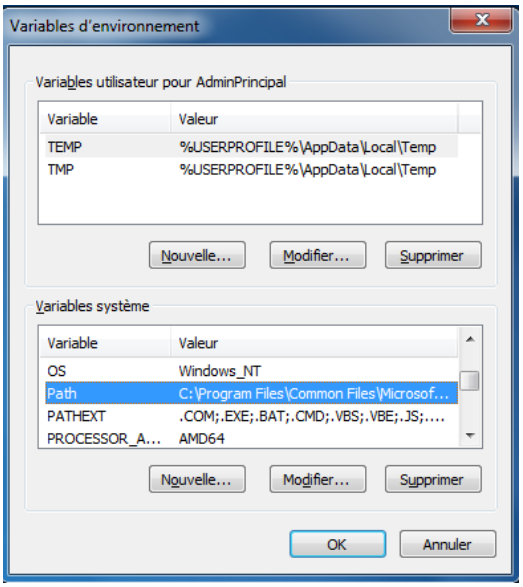
La valeur de la variable peut varier selon la version du programme installée ainsi que l'emplacement des dossiers. Voici les trois variables à ajouter :







Sélectionner ensuite la variable Path, et appuyer sur le bouton « Modifier ».



Voici ce qu'il faut rajouter dans la variable (sans oublier les points virgule) :

- %M2_HOME%\bin;
- %JAVA_HOME%\bin;
- %ANDROID_HOME%\tools;
- %ANDROID_HOME%\plateform-tools;

Pour vérifier que tout fonctionne bien ouvrez une invite de commande et tapez « mvn –v » vous devriez avoir un résultat similaire à ça :

```
D:\Users\AdminPrincipal>mvn -v
Apache Maven 3.0.4 (r1232337; 2012-01-17 09:44:56+0100)
Maven home: C:\Program Files\apache-maven-3.0.4
Java version: 1.7.0_09, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.7.0_09\jre
Default locale: fr_CH, platform encoding: Cp1252
OS name: "windows 7", version: "6.1", arch: "amd64", family: "windows"
```

Si ce n'est pas le cas, revérifier le nom de vos variables d'environnement.

L'étape suivante est d'installer les artefacts Android dans votre dépôt Maven. Pour cela il existe un outil appelé Maven Android SDK deployer. Téléchargez-le à l'adresse suivante :

https://github.com/mosabua/maven-android-sdk-deployer

Décompressez le programme et lancez une invite de commande pour vous rendre sur le répertoire de celui-ci, puis tapez la commande `mvn clean install` si vous souhaitez installer tous les SDK ou alors lancez la commande `mvn install –P 4.1` pour installer la version 4.1 par exemple. Si le message

BUILD FAILURE apparait c'est certainement parce que vous n'avez pas téléchargé tous les éléments nécessaires dans le SDK Manager (exemple Extras). Il est également possible de n'installer qu'un module spécifique, (pour cela référez-vous à la documentation de Maven Android SDK deployer).

CREER UN PROJET ANDROID MAVEN :

Créer un projet Android normal, puis ajouter un fichier POM.XML. Celui-ci doit contenir, les informations classiques d'un fichier POM.XML :

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
         http://maven.apache.org/xsd/maven-4.0.0.xsd">

        <modelVersion>4.0.0</modelVersion>

</project>
```

Ainsi que les dépendances Android, (remplacer le no de version par la version Android utilisée dans le projet):

```
<dependency>
        <groupId>com.google.android</groupId>
        <artifactId>android</artifactId>
        <version>1.5_r4</version>
        <scope>provided</scope>
</dependency>
```

EXÉCUTER UN PROJET ANDROID MAVEN :

Vous pouvez installer un projet Maven Directement dans votre Emulateur (AVD), sans passer par Eclipse. Pour cela :

1. Naviguez jusqu'au répertoire de l'application que vous souhaitez installer

   ```
   $ cd (nom de l'application)
   ```

2. Construisez l'application

   ```
   $ mvn clean install
   ```

3. Lancez l'émulateur (vous pouvez aussi le lancer manuellement).

   ```
   $ mvn android:emulator-start
   ```

4. Installez l'application sur l'émulateur

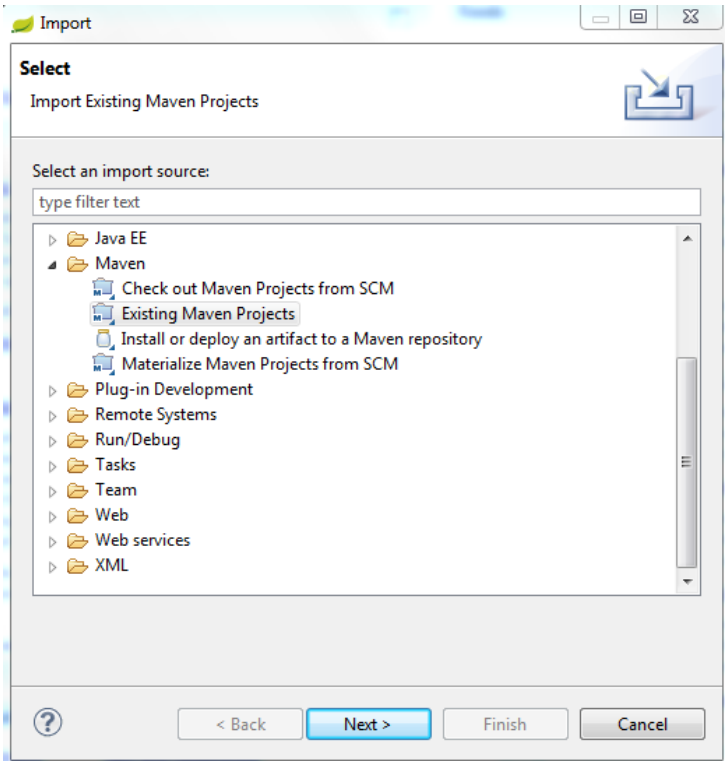
   ```
   $ mvn android:deploy
   ```

5. Lancez l'application

   ```
   $ mvn android:run
   ```

IMPORTER UN PROJET MAVEN ANDROID DANS ECLIPSE SPRING TOOL SUITE :

Dans import au lieu de cliquer sur « Existing Android Code into Workspace », cliquez sur « Existing Maven Projects. » puis sur Next, eclipse va automatiquement télécharger les librairies nécessaires dans votre repository.



Annexe 2
Configuration MAVEN fichier POM.XML

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.springframework.android</groupId>
    <artifactId>spring-android-showcase-client</artifactId>
    <version>1.0.0.BUILD-SNAPSHOT</version>
    <packaging>apk</packaging>
    <name>spring-android-showcase-client</name>
    <url>http://www.springsource.org</url>
    <organization>
        <name>SpringSource</name>
        <url>http://www.springsource.org</url>
    </organization>

    <properties>
        <android-platform>15</android-platform>
        <android-maven-plugin-version>3.2.0</android-maven-plugin-version>
        <maven-compiler-plugin-version>2.3.2</maven-compiler-plugin-version>
        <java-version>1.6</java-version>
        <maven-eclipse-plugin-version>2.8</maven-eclipse-plugin-version>
        <com.google.android-version>4.0.1.2</com.google.android-version>
        <!-- Available Android versions: 1.5_r3, 1.5_r4, 1.6_r2, 2.1.2, 2.1_r1, 2.2.1, 2.3.1, 2.3.3,
4.0.1.2 -->
        <org.springframework.android-version>1.0.0.BUILD-SNAPSHOT</org.springframework.android-
version>
        <org.springframework.social-version>1.0.2.RELEASE</org.springframework.social-version>
        <org.springframework.social-facebook-version>1.0.1.RELEASE</org.springframework.social-
facebook-version>
        <org.springframework.security-version>3.1.0.RELEASE</org.springframework.security-version>
        <org.codehaus.jackson-version>1.9.7</org.codehaus.jackson-version>
        <com.google.code.gson-version>2.2.1</com.google.code.gson-version>
        <org.simpleframework-version>2.6.2</org.simpleframework-version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>com.google.android</groupId>
            <artifactId>android</artifactId>
            <version>${com.google.android-version}</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.android</groupId>
            <artifactId>spring-android-rest-template</artifactId>
            <version>${org.springframework.android-version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.android</groupId>
            <artifactId>spring-android-auth</artifactId>
            <version>${org.springframework.android-version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-crypto</artifactId>
            <version>${org.springframework.security-version}</version>
            <exclusions>
                <!-- Exclude in favor of Spring Android Core -->
                <exclusion>
                    <artifactId>spring-core</artifactId>
                    <groupId>org.springframework</groupId>
                </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>org.springframework.social</groupId>
            <artifactId>spring-social-core</artifactId>
            <version>${org.springframework.social-version}</version>
            <exclusions>
                <!-- Exclude in favor of Spring Android RestTemplate -->
                <exclusion>
                    <artifactId>spring-web</artifactId>
                    <groupId>org.springframework</groupId>
                </exclusion>
                <!-- Provided by Android -->
                <exclusion>
                    <artifactId>commons-logging</artifactId>
                    <groupId>commons-logging</groupId>
```

```
            </exclusion>
        </exclusions>
    </dependency>
    <dependency>
        <groupId>org.springframework.social</groupId>
        <artifactId>spring-social-twitter</artifactId>
        <version>${org.springframework.social-version}</version>
        <exclusions>
            <!-- Provided by Android -->
            <exclusion>
                <artifactId>commons-logging</artifactId>
                <groupId>commons-logging</groupId>
            </exclusion>
        </exclusions>
    </dependency>
    <dependency>
        <groupId>org.springframework.social</groupId>
        <artifactId>spring-social-facebook</artifactId>
        <version>${org.springframework.social-facebook-version}</version>
        <exclusions>
            <!-- Provided by Android -->
            <exclusion>
                <artifactId>commons-logging</artifactId>
                <groupId>commons-logging</groupId>
            </exclusion>
        </exclusions>
    </dependency>
    <dependency>
        <!-- Using Jackson for JSON marshaling -->
        <groupId>org.codehaus.jackson</groupId>
        <artifactId>jackson-mapper-asl</artifactId>
        <version>${org.codehaus.jackson-version}</version>
    </dependency>
    <dependency>
        <!-- Using Gson for JSON marshaling -->
        <groupId>com.google.code.gson</groupId>
        <artifactId>gson</artifactId>
        <version>${com.google.code.gson-version}</version>
    </dependency>
    <dependency>
        <!-- Using Simple for XML marshaling -->
        <groupId>org.simpleframework</groupId>
        <artifactId>simple-xml</artifactId>
        <version>${org.simpleframework-version}</version>
        <exclusions>
            <!-- StAX is not available on Android -->
            <exclusion>
                <artifactId>stax</artifactId>
                <groupId>stax</groupId>
            </exclusion>
            <exclusion>
                <artifactId>stax-api</artifactId>
                <groupId>stax</groupId>
            </exclusion>
            <!-- Provided by Android -->
            <exclusion>
                <artifactId>xpp3</artifactId>
                <groupId>xpp3</groupId>
            </exclusion>
        </exclusions>
    </dependency>
</dependencies>

<build>
    <finalName>${project.artifactId}</finalName>
    <sourceDirectory>src</sourceDirectory>
    <plugins>
        <plugin>
            <groupId>com.jayway.maven.plugins.android.generation2</groupId>
            <artifactId>android-maven-plugin</artifactId>
            <version>${android-maven-plugin-version}</version>
            <configuration>
                <sdk>
                    <platform>${android-platform}</platform>
                </sdk>
                <deleteConflictingFiles>true</deleteConflictingFiles>
                <undeployBeforeDeploy>true</undeployBeforeDeploy>
            </configuration>
            <extensions>true</extensions>
        </plugin>
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>${maven-compiler-plugin-version}</version>
            <configuration>
                <source>${java-version}</source>
                <target>${java-version}</target>
            </configuration>
```

```
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-eclipse-plugin</artifactId>
                <version>${maven-eclipse-plugin-version}</version>
                <configuration>
                    <downloadSources>true</downloadSources>
                    <downloadJavadocs>true</downloadJavadocs>
                </configuration>
            </plugin>
        </plugins>
        <pluginManagement>
            <plugins>
                <!--This plugin's configuration is used to store Eclipse m2e settings only. It has no
influence on the Maven build itself. -->
                <plugin>
                    <groupId>org.eclipse.m2e</groupId>
                    <artifactId>lifecycle-mapping</artifactId>
                    <version>1.0.0</version>
                    <configuration>
                        <lifecycleMappingMetadata>
                            <pluginExecutions>
                                <pluginExecution>
                                    <pluginExecutionFilter>

<groupId>com.jayway.maven.plugins.android.generation2</groupId>
                                        <artifactId>android-maven-plugin</artifactId>
                                        <versionRange>[3.1.1,)</versionRange>
                                        <goals>
                                            <goal>proguard</goal>
                                        </goals>
                                    </pluginExecutionFilter>
                                    <action>
                                        <ignore></ignore>
                                    </action>
                                </pluginExecution>
                            </pluginExecutions>
                        </lifecycleMappingMetadata>
                    </configuration>
                </plugin>
            </plugins>
        </pluginManagement>
    </build>

    <repositories>
        <!-- For testing against latest Spring snapshots -->
        <repository>
            <id>springsource-snapshot</id>
            <name>SpringSource Snapshot Repository</name>
            <url>http://repo.springframework.org/snapshot</url>
            <releases>
                <enabled>false</enabled>
            </releases>
            <snapshots>
                <enabled>true</enabled>
            </snapshots>
        </repository>
        <!-- For developing against latest Spring milestones -->
        <repository>
            <id>springsource-milestone</id>
            <name>SpringSpring Milestone Repository</name>
            <url>http://repo.springframework.org/milestone</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
        <!-- For developing against latest Spring releases -->
        <repository>
            <id>springsource-repo</id>
            <name>SpringSpring Repository</name>
            <url>http://repo.springframework.org/release</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>

</project>

</project>
```