Codename One :

« Write once, run anywhere »

Travail de bachelor réalisé en vue de l'obtention du titre de Bachelor HES en Informatique de Gestion

par :

Xavier COSTA

Conseiller au travail de Bachelor :

Peter DAEHNE, Professeur HES

Haute École de Gestion de Genève (HEG-GE)

Informatique de Gestion

Carouge, le 22 Août 2013

Déclaration

Ce travail de diplôme 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 diplôme, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de diplôme, 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 22.08.2013

Xavier COSTA

Remerciements

Tout d'abord de grands remerciements à Monsieur Peter DAEHNE, responsable de mon travail de Bachelor, pour son aide précieuse et pour le temps qu'il m'a consacré lorsque j'en ai eu besoin.

Je voudrais aussi remercier tous les professeurs de la Haute Ecole de Gestion que j'ai pu rencontrer tout au long de cette formation, qui ont partagé leurs connaissances et expériences professionnelles lors de leurs cours.

Résumé

Codename One est un *plug-in* pour *NetBeans* et *Eclipse* sorti récemment. Il permet de développer en *Java*, avec un seul et même code, des applications multiplateformes.

Nous avons tout d'abord étudié le produit, afin de comprendre comment il fonctionne et de déterminer les inconvénients/avantages qu'il présente. Puis nous l'avons comparé à des produits similaires.

Finalement, nous avons mis en œuvre une application avec trois fenêtres, une base de données et un web service, afin d'implémenter les caractéristiques de base de la plupart des applications mobiles. Ceci nous a permis d'évaluer *Codename One* et d'en donner une critique personnelle.

Table des matières

1. Introduction

Mon travail de fin d'études (travail de Bachelor) consiste à étudier *Codename One,* un nouveau plug-in pour les environnements *NetBeans* et *Eclipse* permettant de développer avec un seul et unique code *Java* une application Smartphone multiplateforme. Le principal de mon travail consiste à développer une application qui fonctionnera sur *Android* et *iOS*, qui sont actuellement les deux systèmes d'exploitation Smartphone où le développement d'applications est le plus populaire ; de plus le développement sur ces deux supports est fondamentalement différent. En effet, le développement d'applications pour *Android* s'effectue à partir de la plupart des plateformes existantes en employant le langage *Java* alors qu'une application iOS est obligatoirement développée sur du matériel *Apple* en utilisant le langage propriétaire *Objective-C*. Suite à ce développement je donnerais une évaluation du produit qui finalement est aussi l'intérêt de ce travail.

Pour ce faire je vais vous présenter tout d'abord le produit ; comment il a été crée, son fonctionnement, les services qui sont proposés, une première évaluation sur le produit et le comparer avec des produits similaires déjà existants. Ensuite, nous verrons comment installer le *plug-in*, comment créer une première petite application et comment utiliser le *designer* de fenêtres. Pour la partie mise en œuvre, une application mettant en pratique les fonctionnalités courantes d'applications sera développée. Nous développerons deux versions de l'application : une avec uniquement du code et une autre à l'aide du *designer* de fenêtres. Quelques explications seront ensuite données sur le code de l'application. Finalement, suite à ce développement, nous pourrons suggérer quelques bonnes pratiques et effectuer une critique du produit.

Les principales sources de mes recherches sont le site *CodeName One* qui fournit quelques ressources (descriptifs, exemples de codes, documentations, etc…), les forums communautaires et quelques sites informatiques fournissant des articles sur le sujet. Au vu de la récente apparition de ce plug-in, il n'y a actuellement pas beaucoup d'autres informations le concernant ailleurs que sur la toile.

2. Présentation

2.1.Historique

Le projet a été lancé par les concepteurs de *LWUIT*[1], Chen Fishbein et Shai Almog, en 2012. *LWUIT* avait pour but de réduire la disparité qu'il y avait entre *J2ME*[2] et *BlackBerry OS* en proposant un standard d'interface utilisateur de beaucoup plus haut niveau que la base commune de l'époque. L'idée de *Codename One* est de proposer une interface de développement commune, codée en Java et capable de fonctionner sur *iOS, Android, BlackBerry OS, Windows Phone 7* et *J2ME*. L'objectif principal est de disposer d'une plateforme permettant de développer rapidement et facilement des applications multiplateforme.

Les concepteurs ont repris les bases du concept de développement de *LWUIT* et l'ont adapté au monde du Smartphone actuel en y ajoutant un simulateur et en dérivant de l'abstraction de *LWUIT*. Ils ont rajouté un compilateur sur le Cloud qui construit les applications des différentes plateformes à partir du *Bytecode Java*[3].

2.2.Structure de Codename One

Codename One est structuré plus ou moins de la même façon que l'environnement Java, mais il utilise une approche *SaaS*[4]. Il est composé comme suit :

- Une interface de programmation (API) où on retrouve toutes les bibliothèques *Java* et *Codename One* que l'on peut utiliser.
- Une interface de *design* (GUI) permettant de créer des thèmes et des fenêtres pour les applications.
- L'interface de développement proposant un simulateur avec la possibilité de visionner l'aspect de l'application sur les différents systèmes d'exploitation grâce à des skins et de tester le fonctionnement de l'application sur l'IDE[5].

---

[1] **LightWeight UI Toolkit (LWUIT) :** Bibliothèque open source développée par Sun Microsystems, conçue spécifiquement pour le développement d'applications mobiles. *Source Wikipedia* http://fr.wikipedia.org/wiki/LWUIT

[2] **J2ME :** est le Framework *Java* spécialisé dans les applications embarquées. *Source Wikipedia* http://fr.wikipedia.org/wiki/Java_Platform,_Micro_Edition

[3] **Bytecode Java :** Code exécutable pour la machine virtuelle Java

[4] **Software as a service (SaaS):** Les clients ne paient pas de licence d'utilisation pour une version, mais utilisent généralement gratuitement le service en ligne ou payent un abonnement récurrent. *Source Wikipedia* *http://fr.wikipedia.org/wiki/Logiciel_en_tant_que_service*

[5] **Integrated development environment (IDE) :** Interface de développement

- La construction des applications (Build) ne se fait pas sur l'IDE mais dans le Cloud sur des serveurs *Codename One* ; il faut ensuite aller récupérer le résultat de la construction sur son compte *Codename One*.

Actuellement sur *Android*, *BlackBerry OS* et *J2ME* le code *Java* standard est exécuté tel quel. La syntaxe de *Java 5* est traduite de manière transparente à *JDK 1.3* sur *J2ME/BlackBerry OS* afin d'assurer le plus possible la comptabilité sur tous les dispositifs. Ceci est réalisé, entre autres, grâce à *Retroweaver*[6], qui transforme les classes *Java* compilées en 1.5 vers du 1.3 qui peuvent être exécutées sur n'importe quel machine virtuelle compatible 1.3.

Actuellement pour *iOS*, *Codename One* utilise la librairie *XMLVM*[7], qui reçoit du code *Java Bytecode* et le recompile dans le langage *iOS*, afin de générer du code natif.

Pour les *Windows Phone*, un traducteur *C#* est utilisé, mais apparemment d'autres solutions sont à l'étude actuellement.

2.3.L'offre

La plus-value de *Codename One* est bien entendu son compilateur d'applications, qui est situé sur le Cloud ; l'utilisateur n'y a pas accès directement. La principale raison est bien entendu financière : cela permet de vendre ce service aux différents utilisateurs via différents abonnements, mais aussi de mieux protéger le produit. Sans abonnement, il n'est pas possible de « *builder* » les applications sur le Cloud mais on peut très bien développer et simuler des applications sans. *Codename One* offre la possibilité de souscrire à quatre abonnements différents (les prix indiqués sont mensuels) :

- **Free** (0$) : Cet abonnement gratuit est très limité. Vous pourrez compiler vos applications, mais le nombre de compilations reste limité. Cet abonnement est conseillé si vous voulez juste essayer le plug-in et effectuer quelques compilations.
- **Basic** (9$) : Cette version propose principalement, en plus de la version **Free**, un nombre illimité de compilations, mais ne permet pas la

[6] **Retroweaver :** permettant de convertir le fichier *Java class* compilé sous 1.5, en fichier *class* qui peuvent être lu sur des anciennes machines virtuelles. http://retroweaver.sourceforge.net/

[7] **XMLVM :** Framework flexible pour la compilation multiplateforme, au lieu de convertir le code source de haut niveau des langages de programmation, il traduit les instructions du bytecode. http://xmlvm.org/

compilation simultanée (Concurrent Builds). Cette version est suffisante si vous l'envisagez pour une utilisation personnelle.

- **Pro** (79$) : L'intérêt principal de cet abonnement est de pouvoir effectuer des compilations simultanées. Si vous devez débugger « *On device* » et tester beaucoup de compilation différentes, au vu du temps de compilation sur le Cloud, il peut être utile de pouvoir envoyer plusieurs compilations en même temps. Vous avez aussi la possibilité d'avoir une assistance par mail. Ce compte est conseillé si on développe souvent et régulièrement sous *Codename One* ; le fait de pouvoir envoyer plusieurs compilations simultanément permettra de gagner du temps.
- **Enterprise** (399$) : Par rapport à l'abonnement, **Pro** il apporte surtout plusieurs services d'assistance supplémentaires.

Le prix de toutes les licences présentées s'entend pour un seul et unique poste de travail. Pour l'acquisition de plusieurs licences, il est recommandé de prendre contact avec le service client afin de négocier un prix.

Il faut aussi savoir que les compilations sur Cloud pour les abonnements **Pro** et **Enterprise** sont prioritaires dans la file d'attente. Il peut arriver que la compilation d'une application *iOS* prenne plus de dix minutes avec un compte **Basic** ; il faut donc bien prendre cela en compte dans le choix de son type d'abonnement.

Pour plus de détails sur les différents abonnements consultez le site de *Codename One* : http://www.codenameone.com/pricing.html

2.4.Avantages/Inconvénients

Descriptions	Avantages	Inconvénients
Le simulateur de l'application n'est pas une machine virtuelle de chaque système d'exploitation. Il ne fait que charger le thème natif ou adapter le thème que l'on a défini au skin du Smartphone.	Rapidité pour tester les fonctionnalités et le comportement de l'application.	Si certaines fonctionnalités ne fonctionnent pas sur tous les systèmes d'exploitation, on ne pourra pas s'en apercevoir avant de l'avoir installer sur le Device.
La compilation se fait sur le Cloud et non sur l'IDE.	Peut permettre, si l'on travaille à plusieurs, d'avoir accès aux compilations des autres.	La compilation peut parfois être longue suivant la taille de la file d'attente présente. On doit forcement être relié et envoyer des informations sur le réseau internet, ce qui pour certaines sociétés est un problème de sécurité.
Création/Edition d'une seule interface graphique pour tous les systèmes d'exploitations.	Gain de temps.	Chaque système d'exploitation possède sa propre interface ; si l'on veut obtenir des interfaces graphiques qui ressemblent à quelque chose, il faut créer un thème pour chaque OS. L'interface graphique obtenue n'est pas aussi propre que celle que l'on obtient normalement ; s'il s'agit d'une de vos priorités, il sera mieux de ne pas réaliser votre application au moyen de *Codename One*.
Un seul et unique code pour toutes les plateformes mobiles.	Gain de temps : - Une seule plateforme de développement. - Un seul langage de développement. - Une seule version du code. - Lors de modifications il suffit juste de recompiler pour les différentes cibles.	L'application est restreinte aux fonctionnalités compatibles des plateformes mobiles ciblées.

Certes, *Codename One* a ses défauts, mais ce que l'on souhaite c'est d'avoir une application multiplateforme fonctionnelle, avec un seul et unique code et il le fait très bien.

Il ne faut pas regarder ce que l'on obtiendrait en codant sur chaque plateforme l'application, mais le gain de temps que *Codename One* apporte. Le *plug-in* n'en est qu'à ses prémices, il faut s'attendre à des améliorations futures.

2.5.Limites

La principale limite de *Codename One* est la compatibilité de certaines fonctionnalités sur les différentes plateformes. Par exemple, il n'est pas possible d'utiliser *SQLite* sur tous les supports ; si votre cible est *Android* et *iOS* cela ne posera pas de problème, mais pour les autres systèmes d'exploitation la portabilité ne sera pas forcement assurée.

2.6.Application concurrentes

Ici, nous allons comparer deux autres produits similaires. Ces produits servent à produire du code cross-plateforme, mais ne sont pas du tout conçus de la même manière et fonctionnent bien différemment. Il est bien entendu que les comparaisons sont faites sur la base des descriptions fournies par les éditeurs ; un test exhaustif de ces *Frameworks* est hors du périmètre de ce travail. Mais il était intéressant de savoir ce qu'il se faisait déjà sur le marché et de pouvoir les comparer à *Codename One*. J'ai retenu les deux principaux produits disponibles dans mon analyse : *PhoneGap* et *Appcelerator Titanium*

2.6.1. PhoneGap

La technologie utilisée est basée sur les standards *Web (HTML5, JavaScript et CSS3)* ce qui rend les applications longues à démarrer sur les devices et beaucoup plus lentes, dû à certains choix d'architecture, sur *iOS*. Les applications développées sur *Codename One* n'utilisent pas cette technologie et sont aussi rapides que le code natif développé sur chaque plateforme. L'avantage de cette technologie est qu'elle est compatible avec quasiment toutes les plateformes mobiles.

De plus, les web technologies ont été conçues pour la distribution *HTTP* et pas pour une exécution locale, alors que *Codename One* a été conçu de zéro afin de fonctionner sur n'importe quelle résolution et fournir des outils visuels pouvant le faire. Vu que le code est exécuté via le browser du device avec *PhoneGap*, si il y a un problème avec l'application, il sera difficile d'identifier le problème. *Codename One* est lié de manière statique à l'application ce qui veut dire que vous avez une application stable avec une version identique de *Codename One* sur chaque dispositif. Puisque le code est manipulé par une seule et même entité, il sera beaucoup plus simple de résoudre les problèmes.

Ce que l'on peut voir aussi, c'est que *Codename One* propose un unique environnement de développement permettant de créer les applications pour toutes les plateformes qu'il propose. Ce qui n'est pas le cas de *PhoneGap* ; par exemple, pour développer une application tournant sur *iOS* et *Windows 7,* on sera obligé de posséder un ordinateur tournant sous *OS X* et utiliser *XCode* tout d'abord et ensuite passer sous *Windows pour* créer l'application sous *Visual Studio.* Certes on pourra réutiliser le même code déjà utilisé sur *OS X*, mais cela veut dire que chaque fois que l'on modifie le code, on doit le faire sur chaque environnement ciblé.

2.6.2. Appcelerator Titanium

Cette solution est très proche de *PhoneGap*. La différence que l'ont peu noter est qu'une application est codée uniquement en *JavaScript*. Quand à son fonctionnement, il convertit le code en *JavaScript* natif de chaque plateforme.

A la différence de *PhoneGap*, *Appcelerator* utilise l'apparence native de chaque plateforme, afin de ressembler au mieux aux applications natives. L'inconvénient qu'il a par rapport à *PhoneGap*, est qu'il n'est pas compatible avec beaucoup de plateformes (*Android et iOS* seulement, compatibilité *BlackBerry* pas encore au point).

Nous pouvons remarquer que ce qui fait la force de *Codename One,* par rapport à ces deux concurrents, est qu'on développe une unique application (code et interface) à partir de laquelle on obtient toutes les versions mobiles de l'application. Si l'on doit faire des modifications elles seront directement appliquées à toutes les versions.

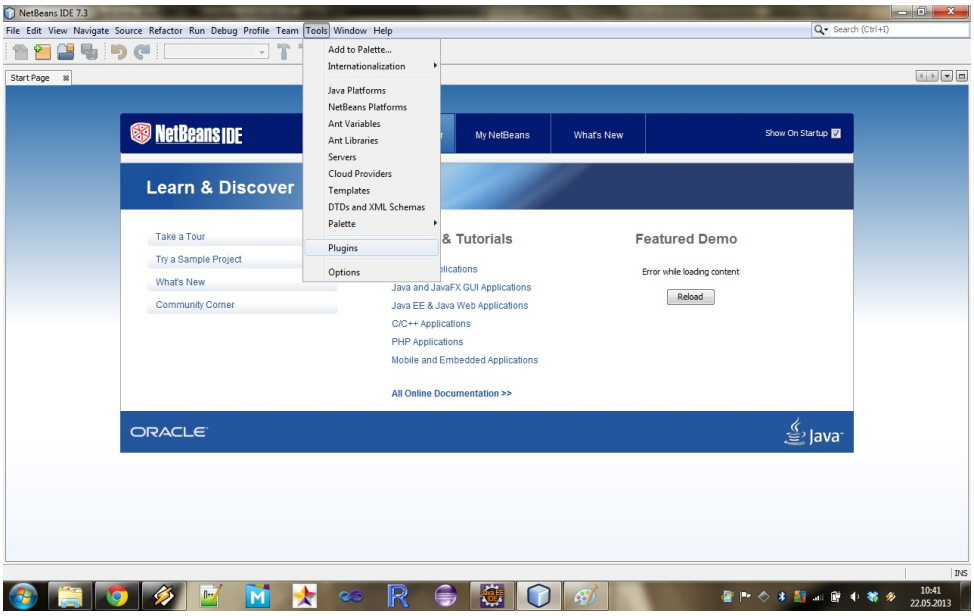
3. Fonctionnement

3.1.Installation

Codename One est un plugin qui va s'ajouter à notre environnement et va nous permettre de créer des projets *Codename One,* afin de développer une application avec le même code pour cinq plateformes mobiles différentes. Il est possible d'installer le plugin *Codename One* sur l'IDE *NetBeans* ou *Eclipse*. Ayant testé préalablement ces deux installations ainsi que l'environnement de développement, mon choix s'est orienté vers *NetBeans*.

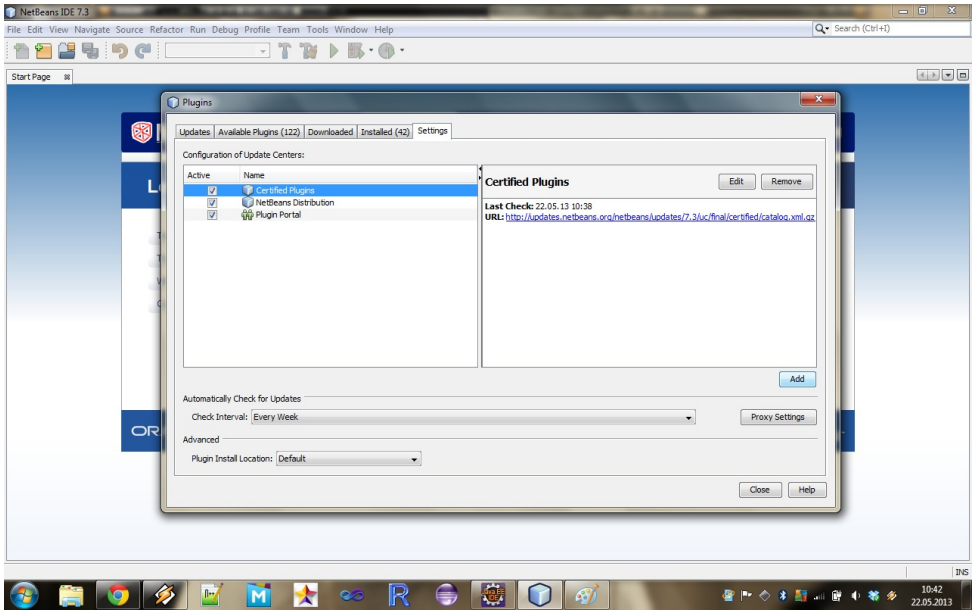
Pour commencer nous allons ouvrir *NetBeans* et aller dans la partie *« plugins »*, qui va nous permettre de rajouter notre interface de développement *Codename One*.

>Tools > Plugins



Nous allons maintenant récupérer le *plug-in* et l'ajouter à notre environnement de développement.

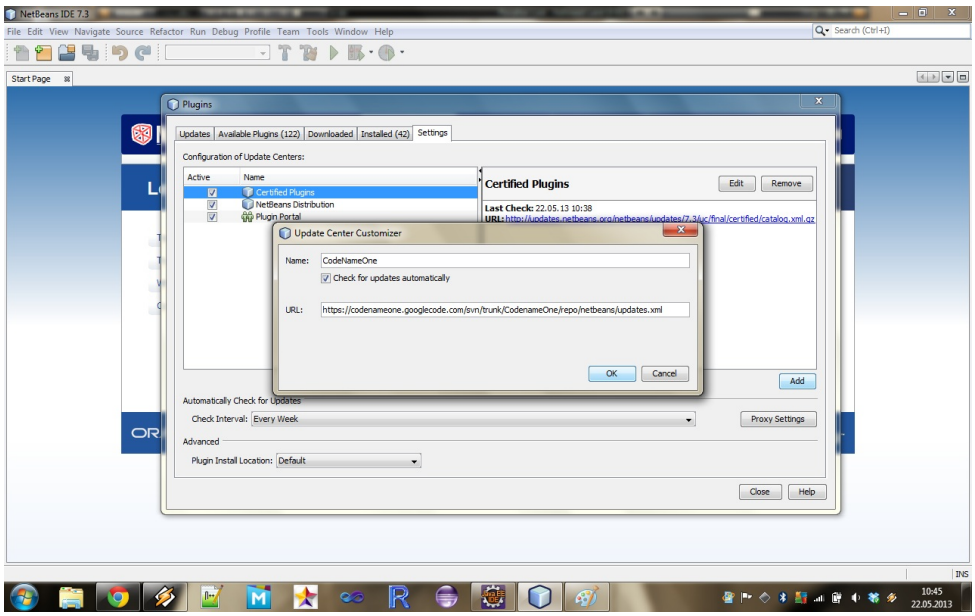
>Add



Une fenêtre va s'ouvrir dans laquelle vous allez devoir choisir un nom pour votre *plug-in*, « *CodeName One* » par exemple, et rentrer l'url ci-dessous afin d'aller récupérer le *plug-in*.

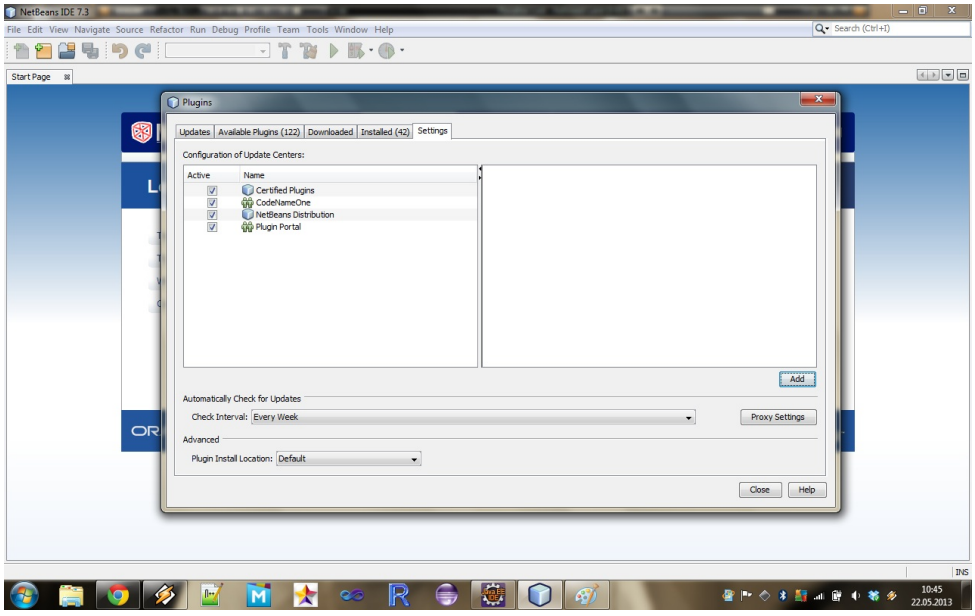
https://codenameone.googlecode.com/svn/trunk/CodenameOne/repo/netbeans/updates.xml

>OK



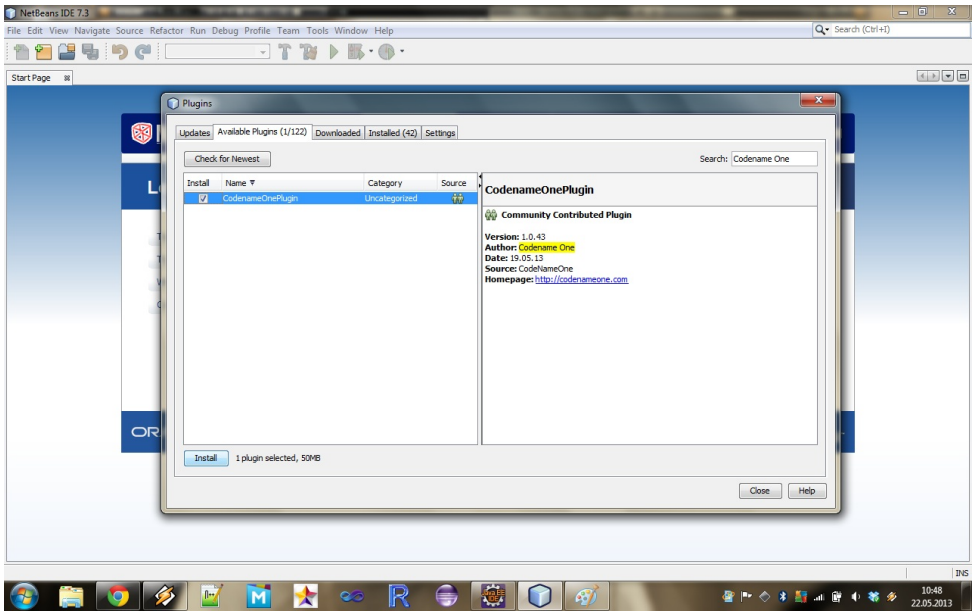
Vous le verrez apparaître comme ci-dessous ; ensuite, nous allons installer le *plug-in* lui-même.

>Available Plugins



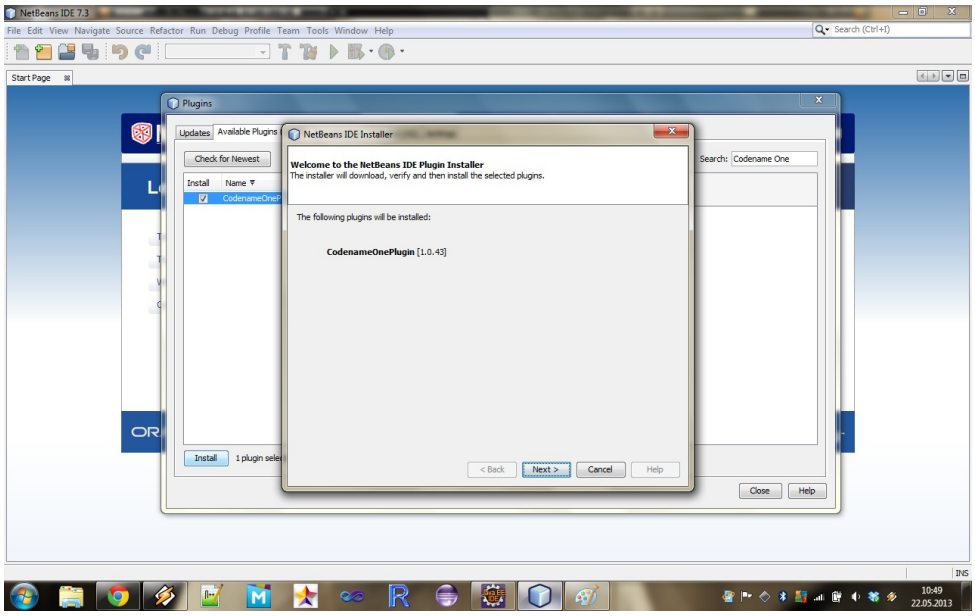
Nous allons procéder maintenant à l'installation ; sélectionnez le *plug-in « Codename One »* et lancez l'installation.

>Install



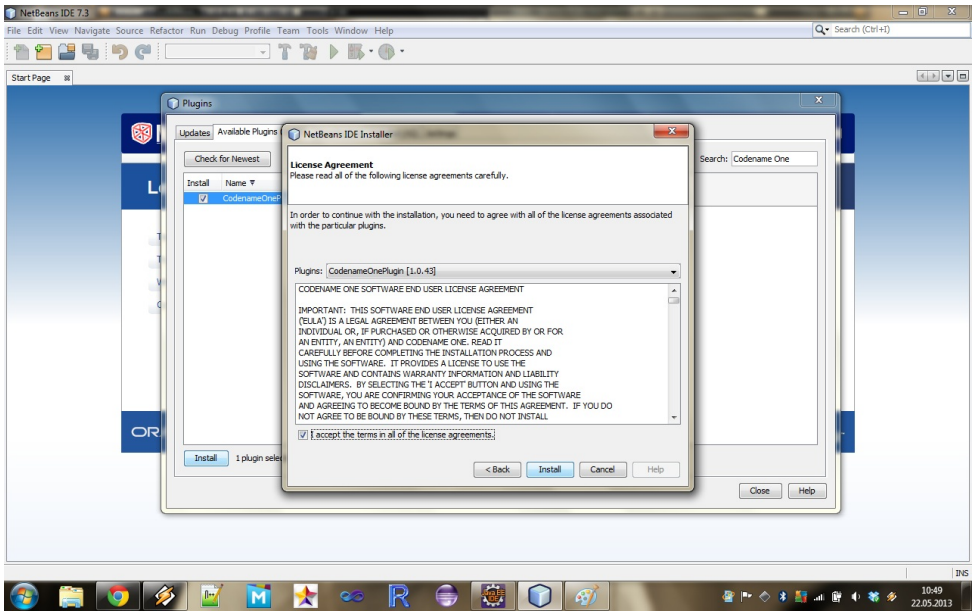
Une fenêtre d'installation vous proposant d'installer le *plug-in* va s'ouvrir, le numéro de version peut différer de celle de l'image. On peut maintenant passer à la suite de l'installation.

>Next

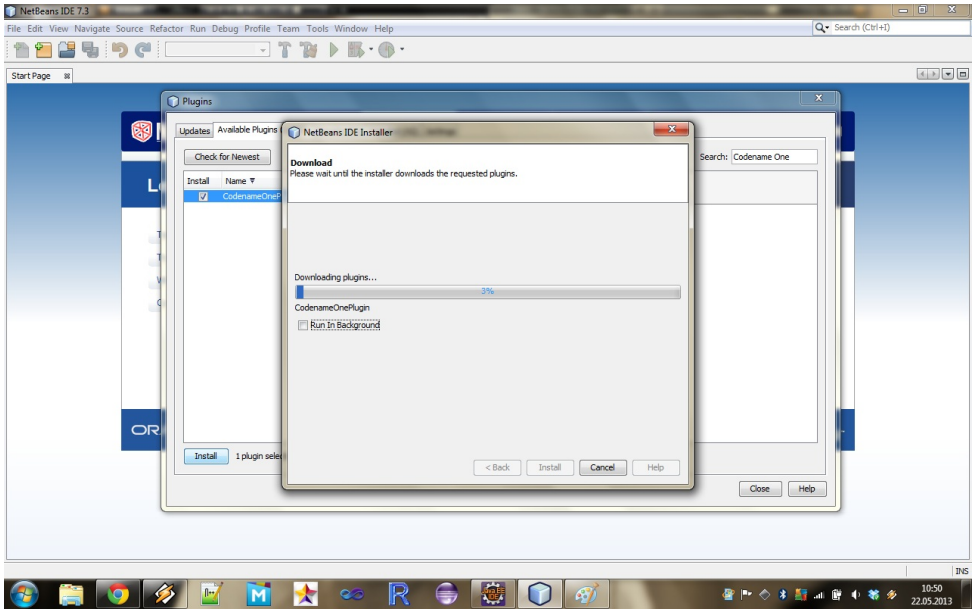


La fenêtre d'acceptation de la licence apparait, acceptez-la et continuez l'installation.

>Install

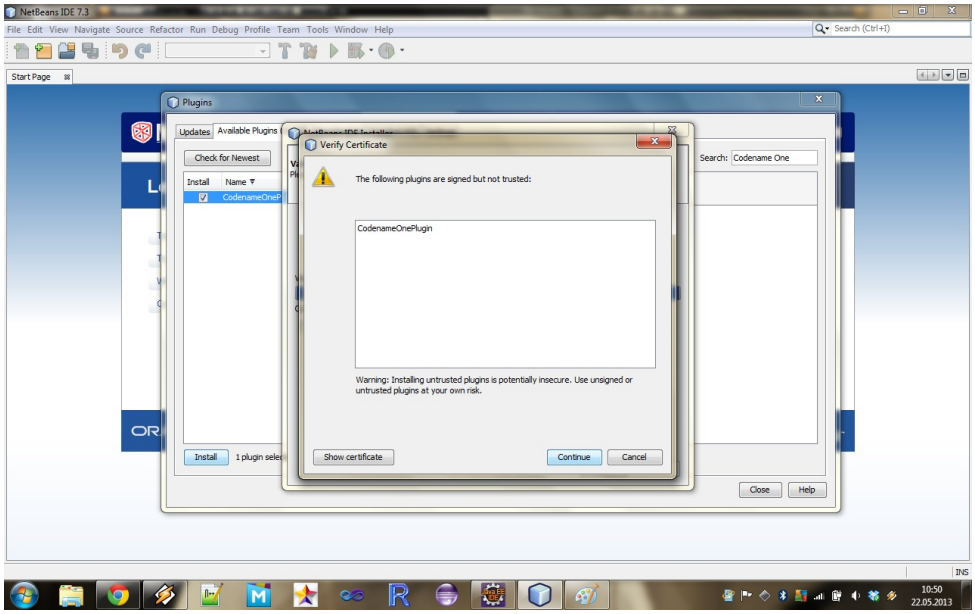


Vous devriez voir apparaître une fenêtre d'installation comme celle-ci, attendez que l'installation se termine.



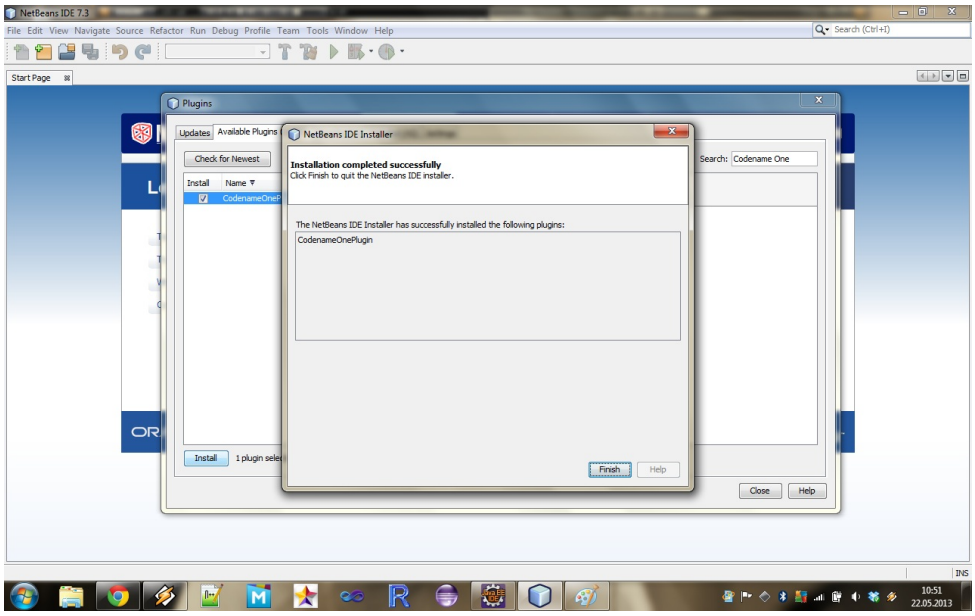
Au cours de l'installation, il se peut que vous ayez un avertissement indiquant que le certificat n'a pas été vérifié et n'est pas sûr. Poursuivez l'installation en appuyant sur le bouton « *Continue* ».

>Continue



Une fois l'installation terminée vous devriez avoir une fenêtre vous avertissant que l'installation s'est déroulée correctement. Vous avez maintenant fini l'installation du *plug-in Codename One*, il vous suffit de relancer *NetBeans*.

>Finish



3.2.Library

Malgré que nous développions en *Java*, beaucoup de librairies seront remplacées par des librairies *Codename One*. Les librairies *Java* disponibles sont :

- java.io
- java.lang
- java.lang.annotation
- java.lang.ref
- java.net (très limité, seules quelques fonctionnalités sont disponibles)
- java.text
- java.util

Quelques exemples de fonctionnalités des librairies *Codename One*:

- Gérer une base de données *SQLite* (com.codename1.db)
- Une interface graphique qui permet de créer des applications avec le « *look & feel* » natif de chaque système d'exploitation. Il est possible, si on le souhaite, de le personnaliser (com.codename1.ui et ses sous-packages)
- Possibilité de lire des fichiers de type *CSV*, *JSON* et *XML* (com.codename1.io et com.codename1.xml)
- Accéder aux contacts du téléphone (com.codename1.contacts)
- Accéder à un service web (com.codename1.io)

Une documentation complète en ligne est disponible, mais elle n'est pas aussi détaillée que la *Javadoc* que l'on connaît :

https://codenameone.googlecode.com/svn/trunk/CodenameOne/javadoc/index.html

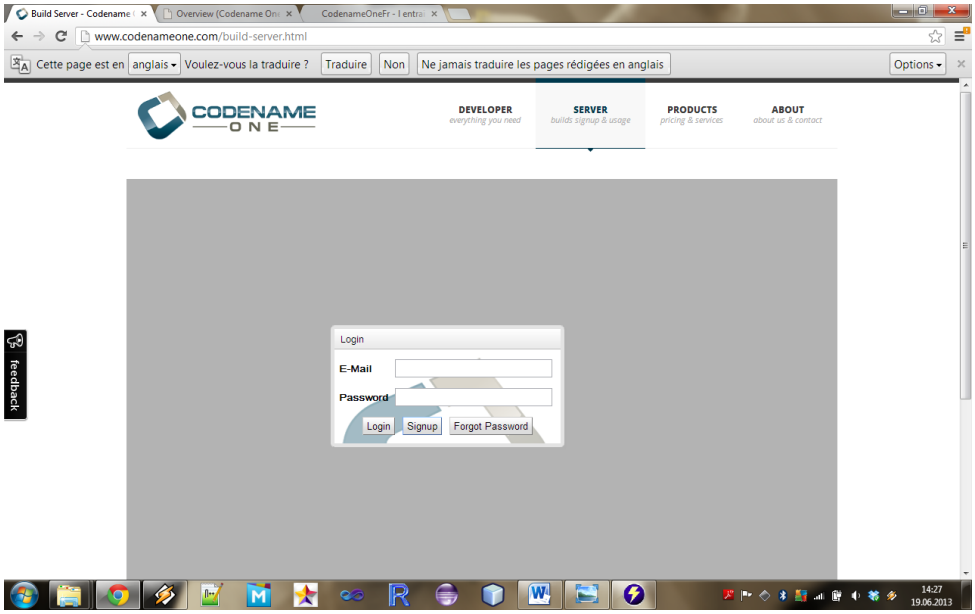
3.3.Cloud

Maintenant que l'installation de l'environnement de développement est terminée, il va falloir vous enregistrer sur le site de *CodeName One* afin de pouvoir compiler les différentes versions de votre application. Les compilations des applications se font sur le Cloud et sont ensuite disponibles sur le compte *CodeName One* que vous avez créé. Il faut aussi tenir compte du type de compte que vous possédez (voir plus haut, chapitre 2.3).

Rendez-vous à l'adresse ci-dessous afin de procéder à l'inscription.

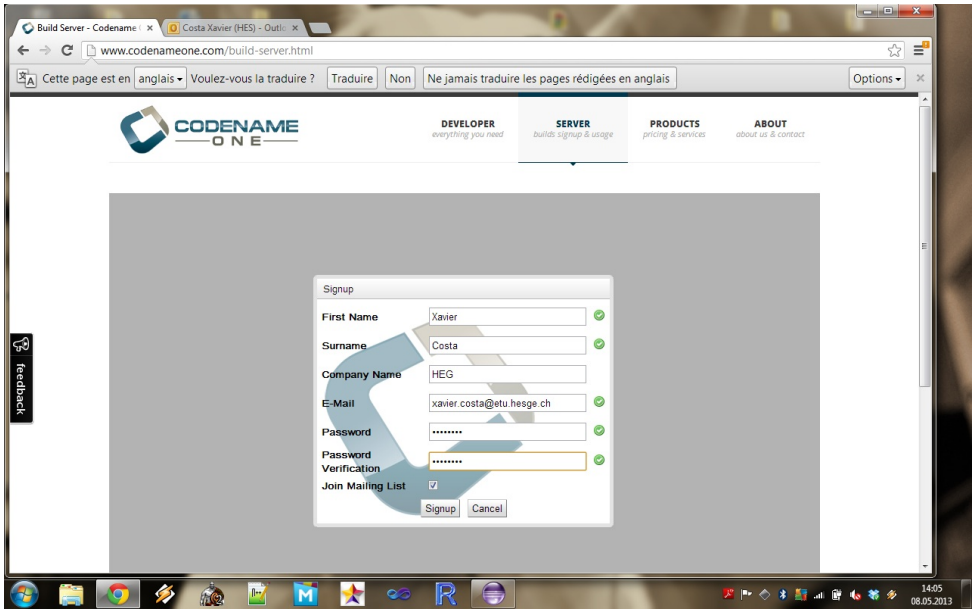
http://www.codenameone.com/build-server.html

>Signup



Rentrez vos informations et terminez l'inscription.

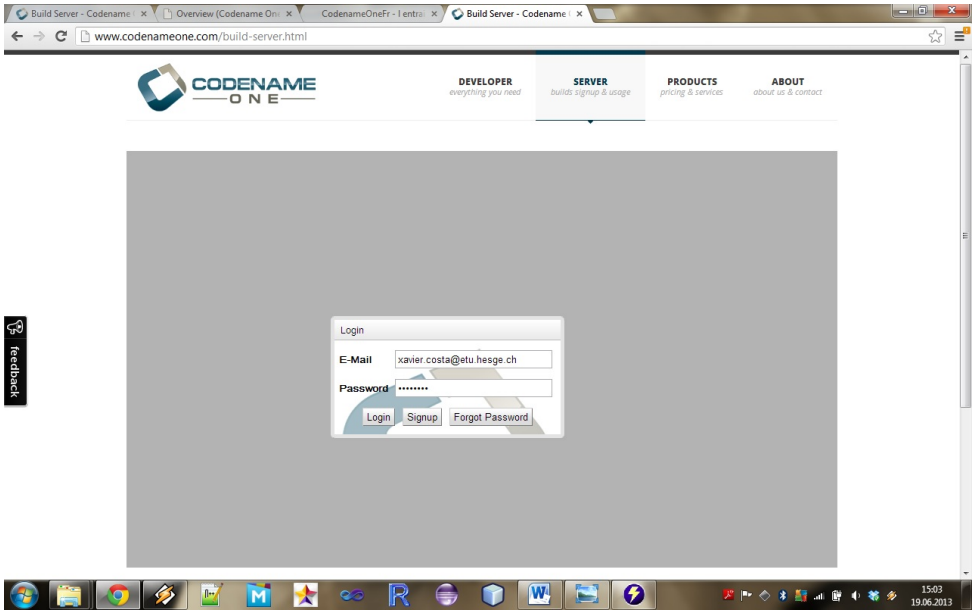
>Signup



Vous pouvez maintenant vous identifier via le même lien que précédemment.

http://www.codenameone.com/build-server.html

>Login

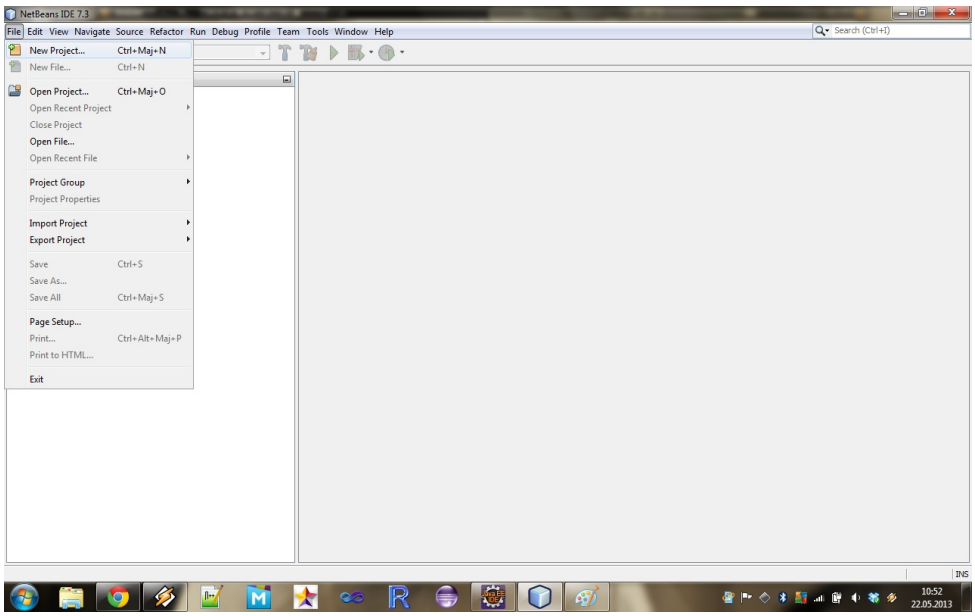


Vous avez maintenant accès à votre compte ; sur l'onglet « *Build Status* » vous aurez accès aux applications que vous compilerez par la suite sous la forme d'un lien de téléchargement ou d'un *QRcode* ; nous verrons cela plus bas. Vous pouvez aussi grâce à l'onglet « *Subscription* » changer votre type de compte si vous le désirez. Il est à noter que la facturation s'effectue exclusivement via un compte *PayPal*.

3.4.Hello World !

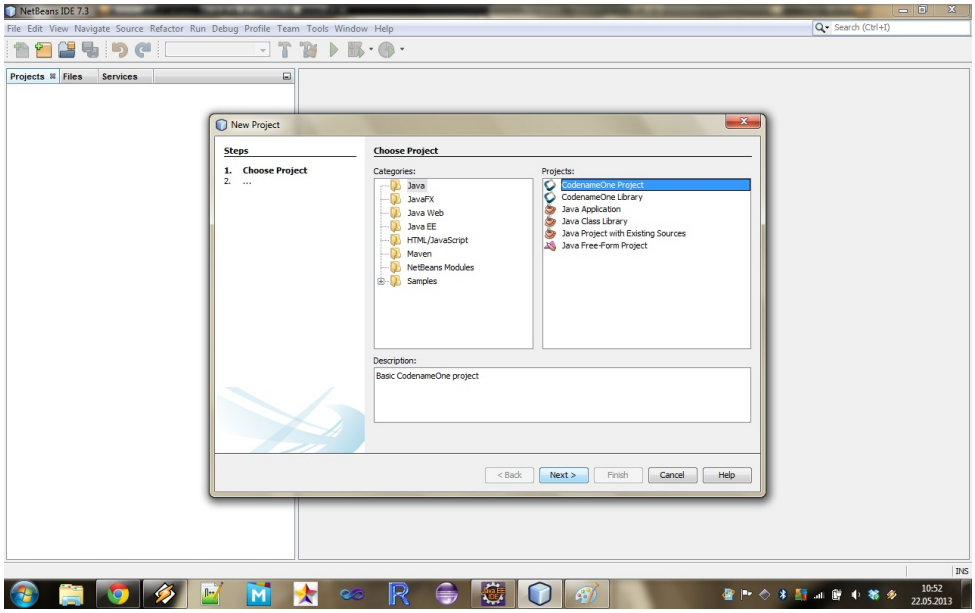
Nous allons maintenant réaliser une application ***Hello World*** pour voir comment on crée une application et comment se déroulent les différentes compilations de celle-ci. Tout d'abord nous allons créer un nouveau projet.

>File>New Project



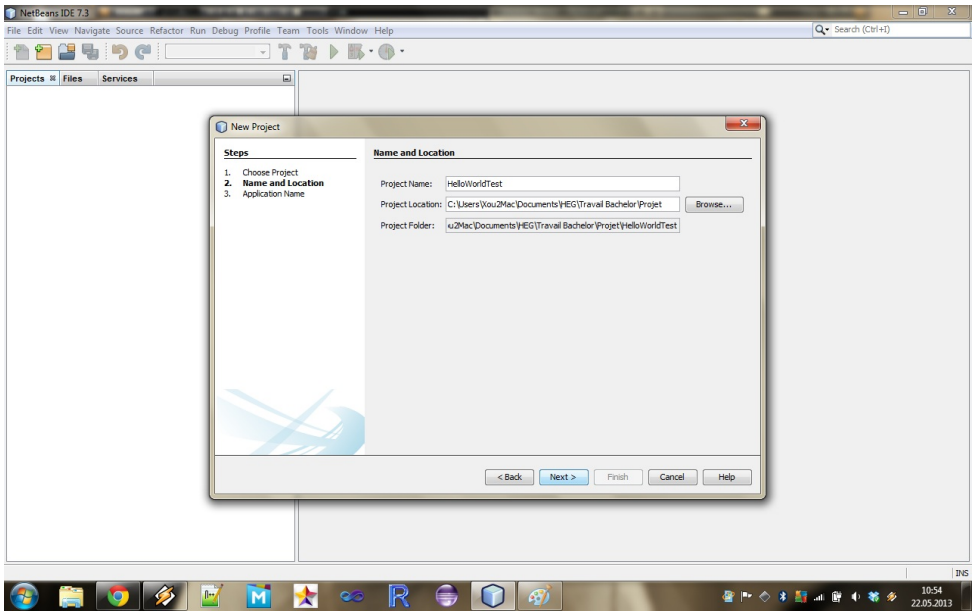
Dans catégorie « *Java* », sélectionnez « *CodenameOne Project* ».

>Next



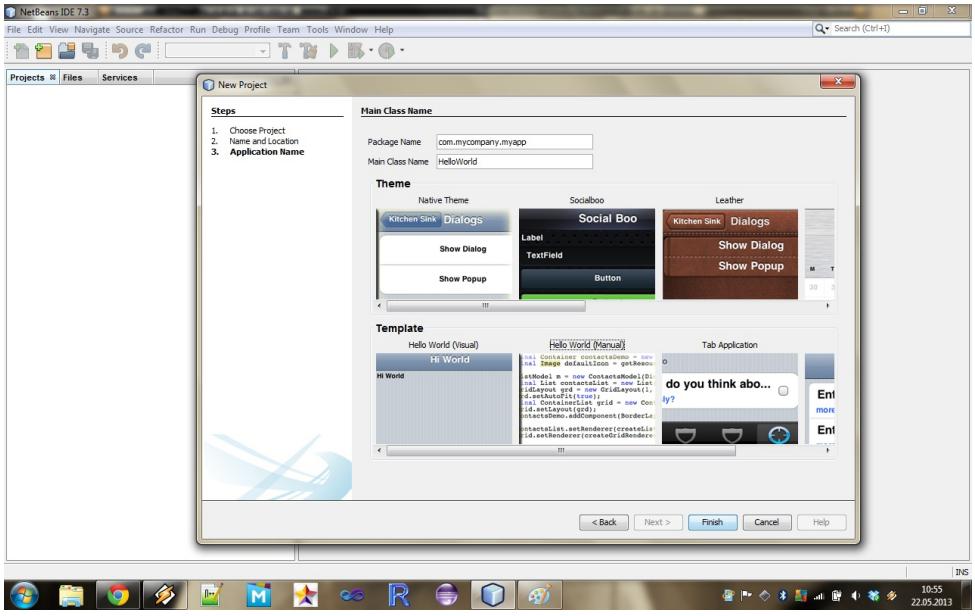
Choisissez le nom de votre projet ainsi que l'endroit où vous voulez l'enregistrer

>Next

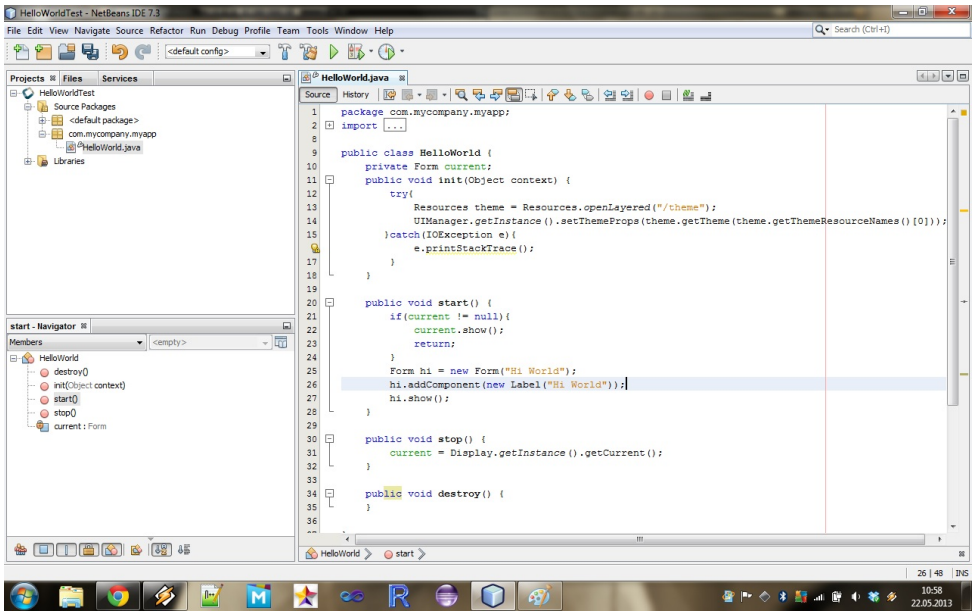


Ici, on va choisir le thème que l'on souhaite ; « *native* » est recommandé. Ainsi, le thème natif de chaque environnement sera pris lors de la compilation. Dans notre exemple, nous allons utiliser l'interface manuelle, sans passer par l'outil de construction d'une interface visuelle. Nous verrons ainsi comment est structuré le code d'une application.

>Finish

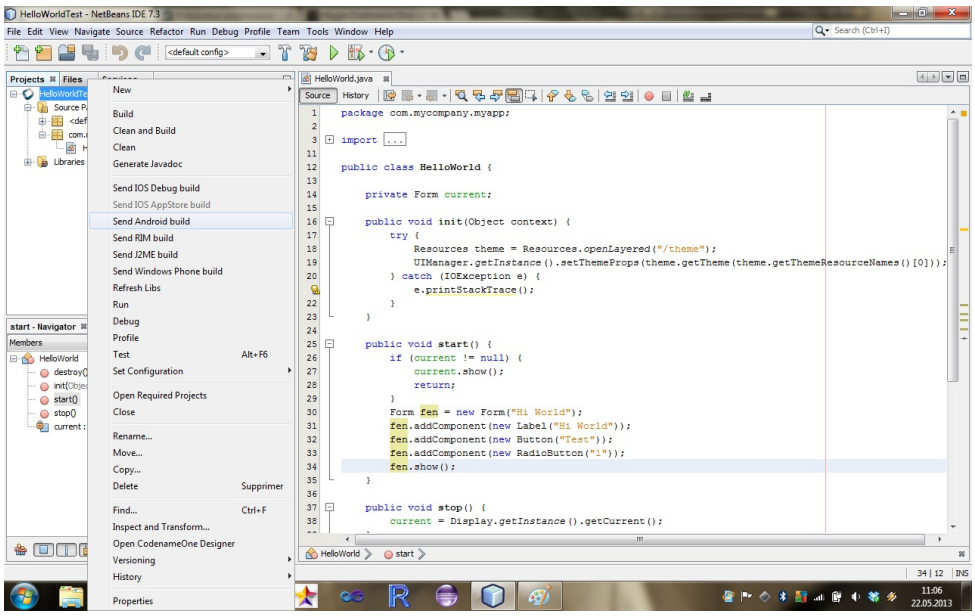


Vous allez obtenir directement le code de votre **Hello World**. On peut voir qu'une application est composée de quatre procédures principales que nous allons examiner de plus près.



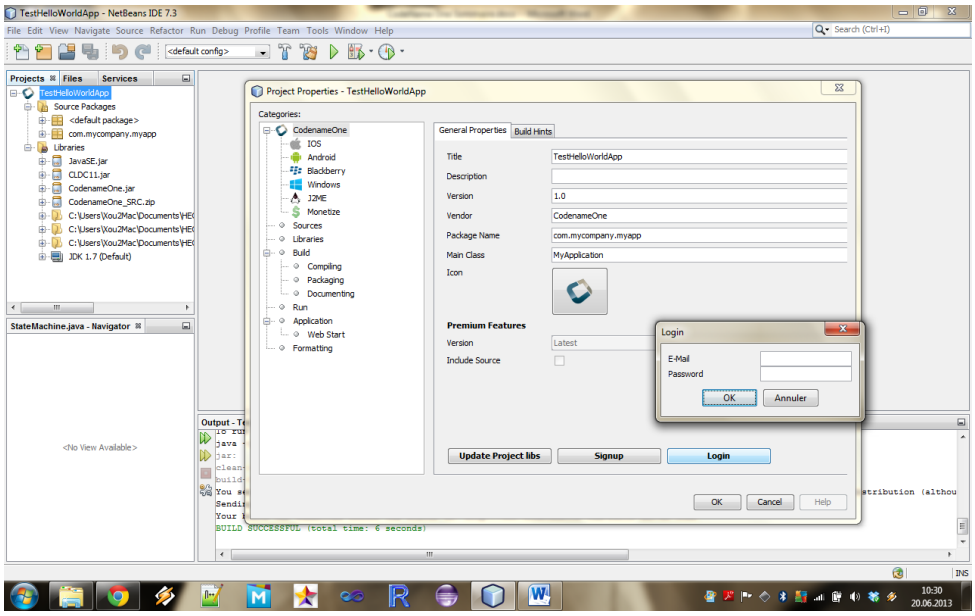
Maintenant que notre application est terminée, nous allons lancer la compilation des versions *Android* et *iOS.* La version *iOS* sera une version pour *iOS Jailbreak,* pour simplifier les choses au niveau du *Market* et actuellement, il n'y a pas de certificat installé permettant de le faire. Sélectionnez votre application et faites un clic droit dessus.

>Send Android Build* et *>Send iOS Debug Build

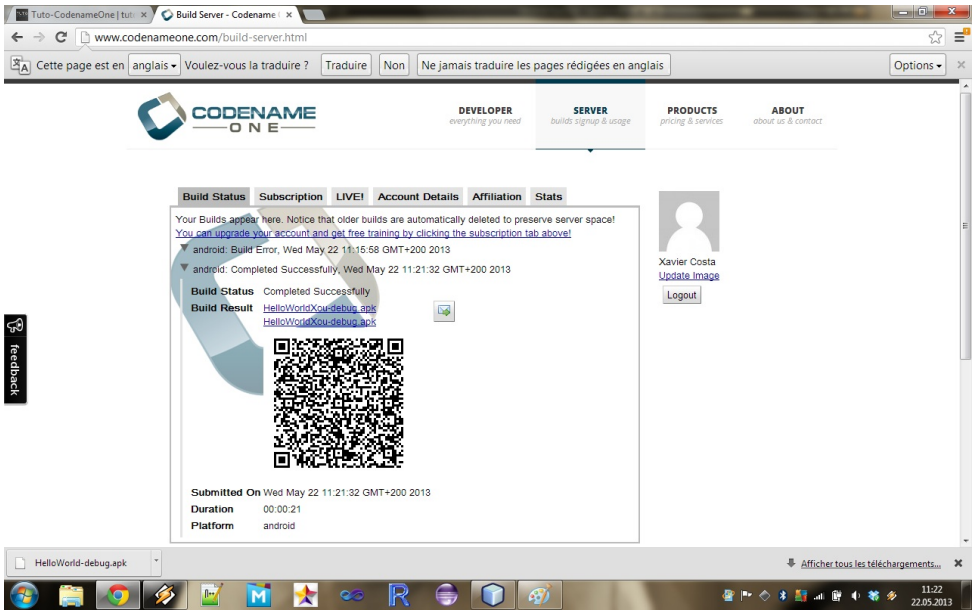


Lors du premier *build*, le système devrait vous demander de vous identifier afin de compiler l'application sur votre compte. Si ce n'est pas le cas, vous pouvez le configurer manuellement en faisant un clic droit sur le projet.

>Set Configuration > Customize...> CodenameOne > Login



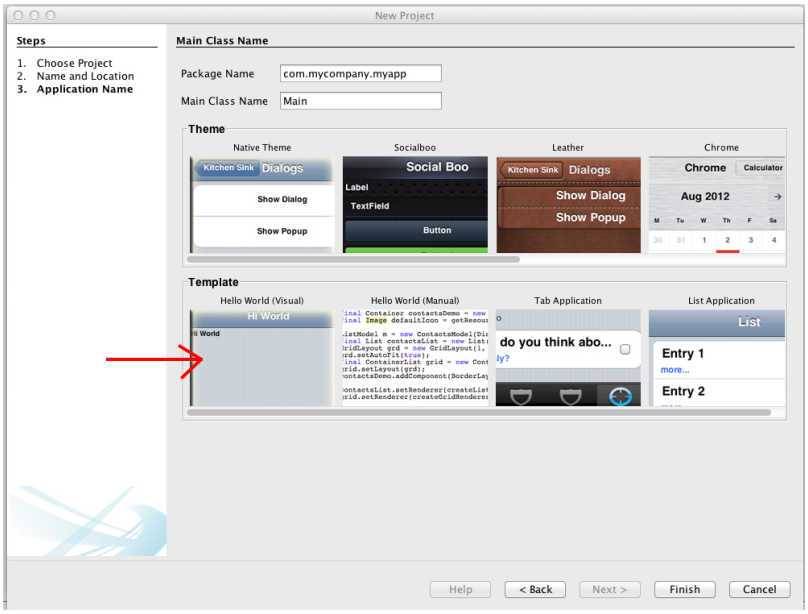
Finalement, retournez sur le site de *Codename One* ; dans votre espace *« BuildStatus »* vous trouverez vos compilations que vous pouvez directement installer sur votre mobile si vous possédez une application *QR code*.



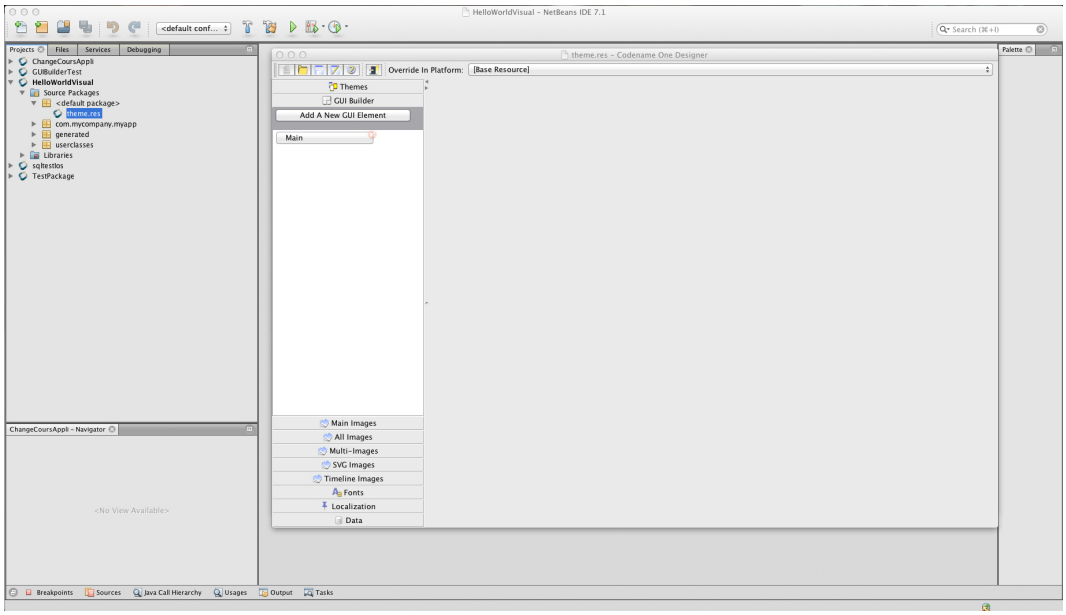
3.5.Interface graphique (GUI Builder)

Codename One propose un éditeur d'interface graphique qui permet de créer des thèmes, créer des fenêtres et ajouter des évènements aux composants. Nous allons tout d'abord regarder ce que l'on peut faire avec cet éditeur et ensuite nous allons créer une application simple qui effectuera la lecture d'un *String* et qui, suite a l'appui sur un bouton, affichera le *String* lu dans une boîte de dialogue (accompagnée d'un texte de formatage).

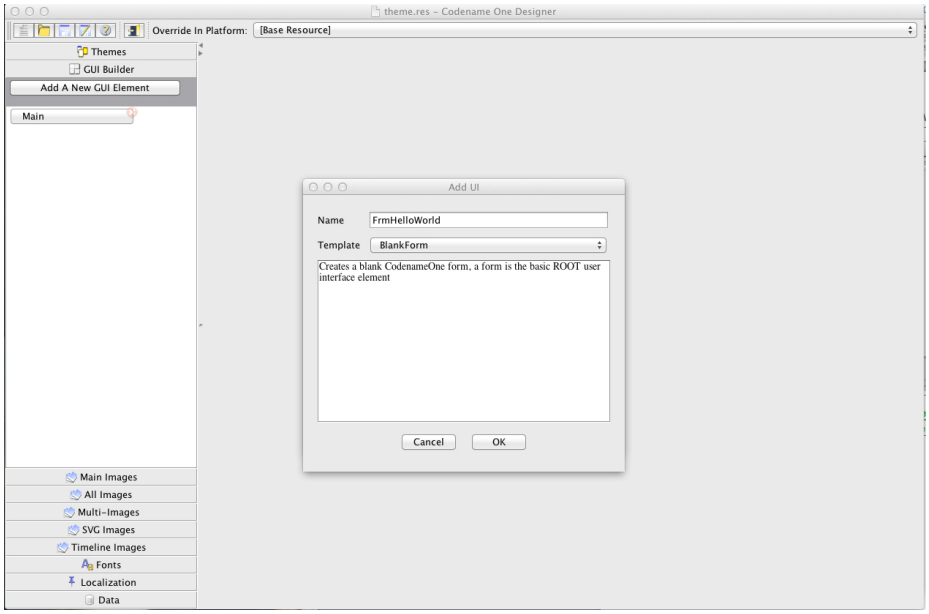
On va recréer un projet comme pour le ***Hello World*** sauf que cette fois nous allons employer le Template « *Visual* » (lors de notre exemple précédent, nous avions utilisé le Template « *Manual* ») ; ceci va permettre de créer automatiquement les machines d'états qui géreront les événements des composants dans lesquels on définira les *listeners* qu'on verra par la suite.



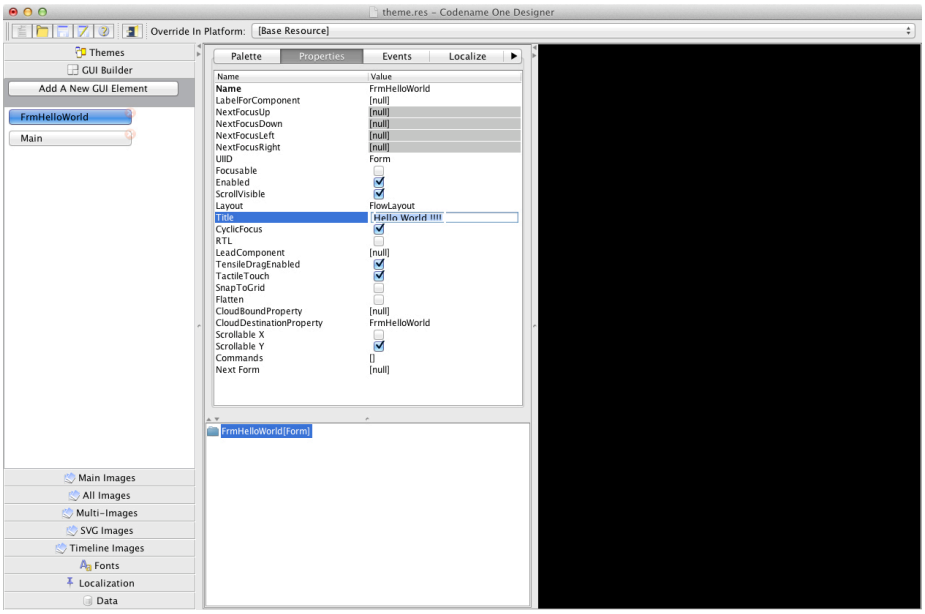
Maintenant, dans les packages de votre dossier, il y a un package « *<default package>* » dans lequel il y un fichier appelé par default « *theme.res* ». Double-cliquez dessus afin d'ouvrir l'éditeur d'interface. Vous devriez avoir cette fenêtre :



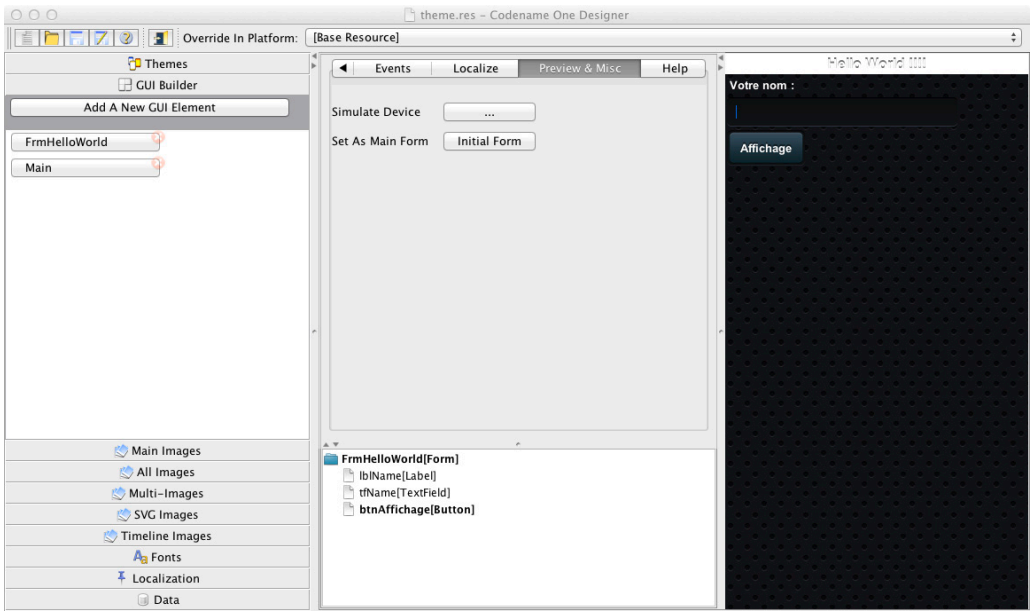
Dans l'onglet *« GUI Builder »*, vous avez toutes les fenêtres créées dans votre projet. Vous pouvez ajouter et créer de nouvelles fenêtres en cliquant sur *« Add A New GUI Element »*, ce que l'on va faire tout de suite. Donnez un nom à votre fenêtre et prenez le template par défaut.



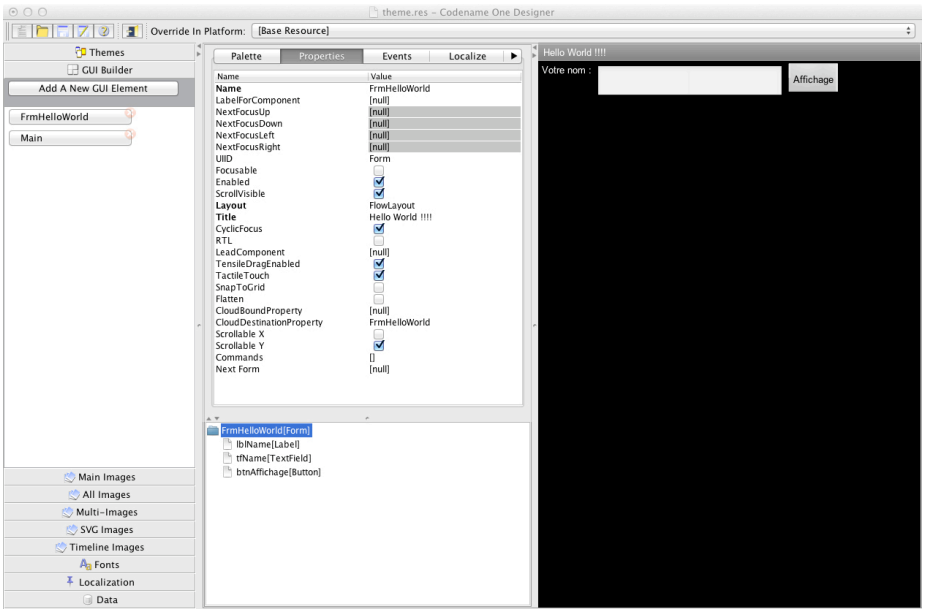
Votre nouvelle fenêtre vide s'ouvre. Il va falloir maintenant ajouter les composants dont on a besoin. Mais tout d'abord, on va donner un titre à notre fenêtre : dans l'arborescence des composants, sélectionnez votre « *Form* » et allez ensuite dans l'onglet *« Properties »* et changez le titre comme sur l'image ci-dessous :



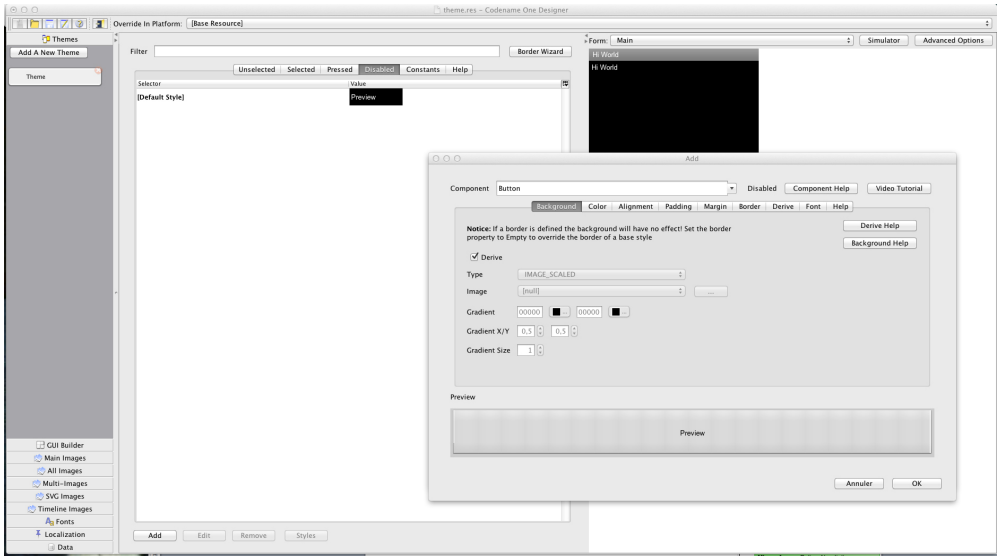
Il faut maintenant définir la fenêtre qui doit s'ouvrir au démarrage de l'application, si jamais vous voulez ajouter d'autres fenêtres. Sélectionnez la « *Form* » que vous voulez mettre en fenêtre de démarrage, puis allez sur l'onglet *« Preview & Misc »* et pour finir cliquez sur *« Initial Form »*.



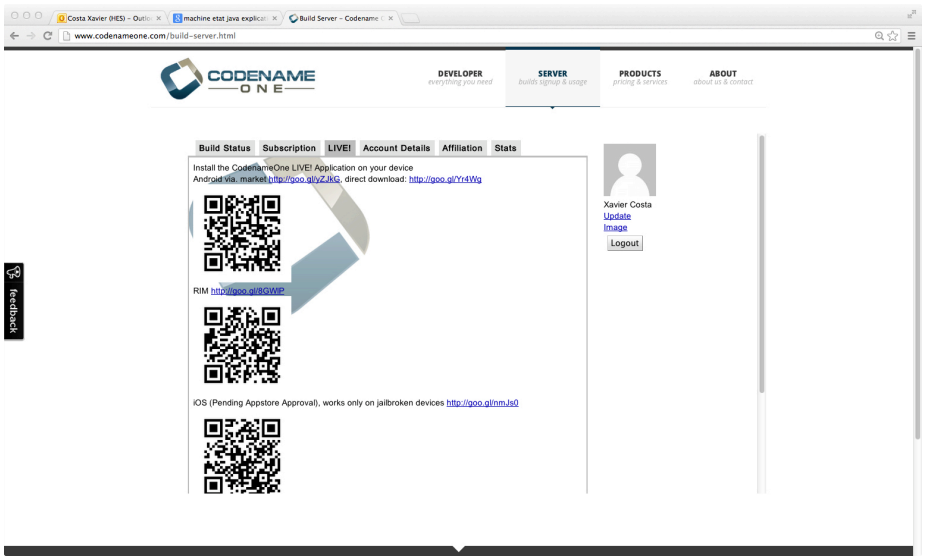
Il est possible de modifier la mise en page des composants de la fenêtre en modifiant les valeurs des propriétés (onglet « *Properties »*). Il faut modifier la propriété « *Layout »* ; par défaut celle-ci a la valeur « *FlowLayout »* (les composant sont mis les uns à la suite des autres en fonction de leurs tailles et de la résolution du *device*). Ici on va garder celui par défaut ; ajoutez un *Label*, un *Textfield* et un *Button* depuis l'onglet « *Palette* »*,* renommez-les ensuite dans l'onglet « *Properties* » et modifiez le texte afin d'obtenir ceci :



Si vous le souhaitez, il est possible de modifier le thème de votre fichier de ressources «*.res »,* par l'entremise de l'onglet « *Themes* ». Vous pouvez définir pour chaque état d'un composant une skin particulière si vous le souhaitez. Pour cela choisissez l'onglet de l'état que vous voulez personnaliser, par exemple « *disable* » et cliquez sur « *add* ». Une fenêtre s'ouvre et vous pouvez choisir le composant à personnaliser : « *Button* » dans notre cas. Vous pouvez ensuite le personnaliser comme vous le souhaitez dans cette fenêtre.



Codename One met à disposition une application qui permet de visualiser en temps réel, sur l'appareil cible, le design de l'application en même temps que vous le créez : *Codename One LIVE!* Vous pouvez la télécharger sur la page de votre compte *Codename One* dans l'onglet *« LIVE! »* (Attention, elle ne fonctionne sur *iOS* que si l'appareil est *Jailbreaké*).

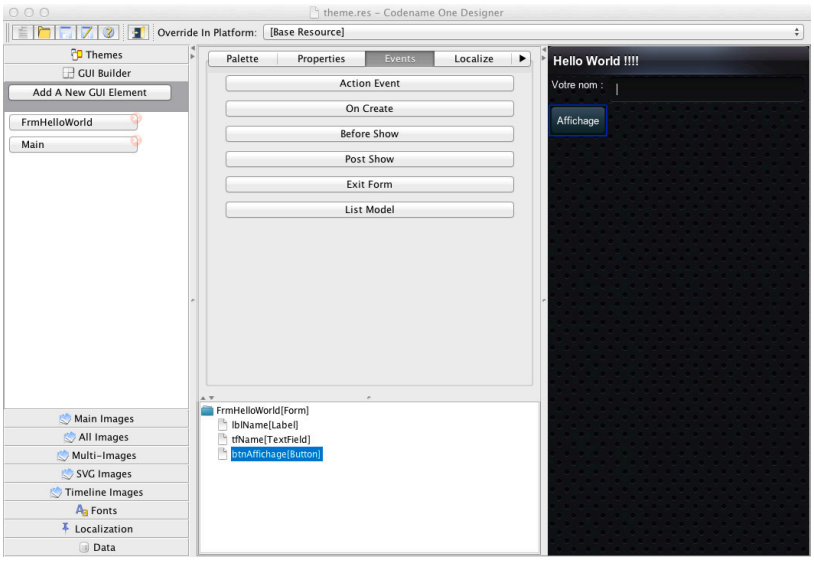


Une fois l'application installée sur votre mobile, assurez vous que *« Live preview »* est activé dans le menu « *Codename One* ».

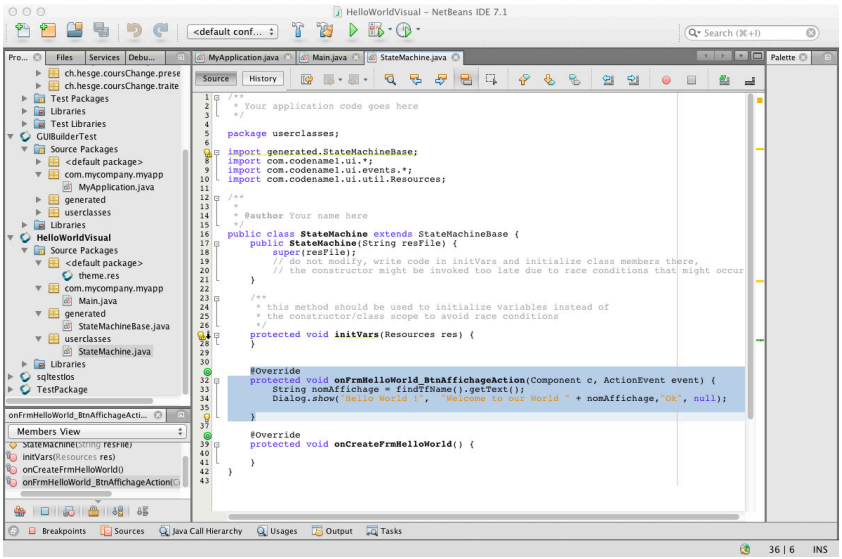
Ouvrez l'application, l'application va vous demander de vous identifier et vous vous retrouvez sur la page de vos *builds*.

Ensuite cliquez sur *« LIVE! »* pour avoir accès aux fenêtres du designer en cours et sélectionnez celui que vous voulez visionner. La fenêtre de la dernière sauvegarde que vous avez faite s'affiche. Essayez de faire une modification, par exemple changer le thème, sauvegarder et normalement la modification apparaît sur votre device.

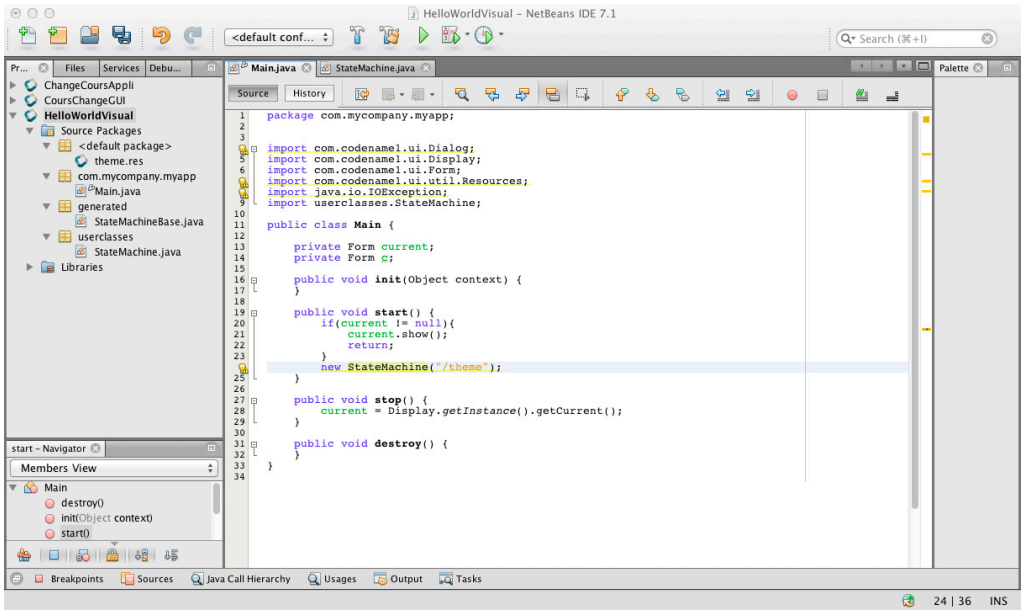
Maintenant, on va ajouter un évènement sur notre bouton afin de déclencher une fenêtre, avec un message affichant le nom rentré. Sélectionnez le bouton et dans l'onglet *« Events »* cliquez sur *« Action Event ».* L'événement est ajouté comme méthode dans une classe de type ***StateMachine***, dans votre projet.



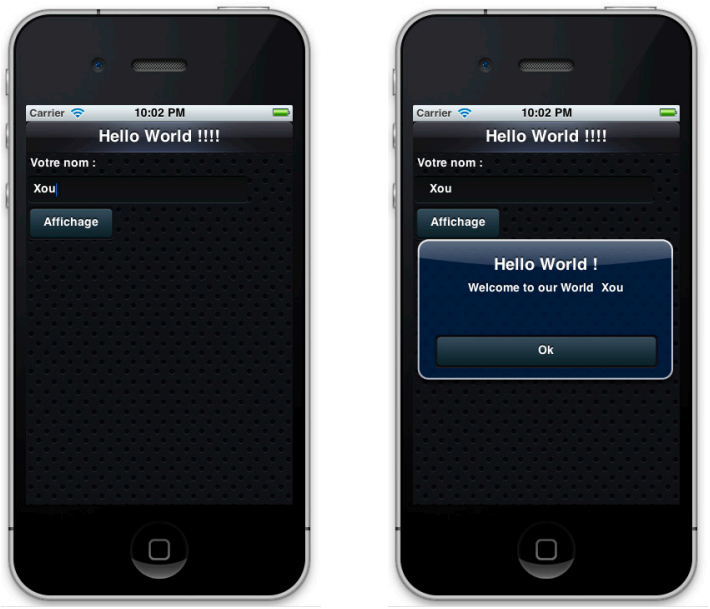
La classe ***StateMachine***, modélisant la machine d'état, se trouve dans le package *« userclasses »*, la méthode ajoutée est nommée automatiquement « *onNomForm_NomComposantAction* » pour les *Action Events.* Il suffit de mettre le code que vous voulez exécuter dans cette méthode. La méthode « *findNameComposant* » est une méthode héritée de ***StateMachineBase****,* qui permet de récupérer tous les composants qui ont été créés dans le fichier de ressources (lorsque vous ajoutez des composants et sauvegardez dans l'éditeur de création de *Form,* la méthode est ajoutée dans ***StateMachineBase***).



Ensuite, il ne vous reste plus qu'à instancier la ***StateMachine*** avec le thème que vous souhaitez dans votre classe ***Main*** comme ci-dessous. Par défaut, *Codename One* crée ce code, mais vous pouvez le modifier, par exemple si vous voulez utiliser un autre fichier de ressource.



Vous obtenez finalement ce résultat si vous avez bien suivi toutes les étapes.



Ce que l'on peut retenir de l'architecture du « *GUI Builder* » est qu'il utilise le pattern *State* pour la gestion des évènements. Ce qui n'était pas le cas lorsque l'on codait directement l'interface dans le code, où l'on utilisait le pattern *Observable* avec des *listeners.*

L'inconvénient du *« GUI Builder »* est, que tout le code des évènements se retrouve dans une seule classe ***StateMachine***, avec aucune différenciation entre les fenêtres. Il est donc beaucoup plus difficile de maintenir le code de l'interface.

Cette situation a pour inconvénient majeur que tout le code de la couche affichage se trouve dans une seule classe, même si l'application comprend plusieurs fenêtres. Il est donc plus difficile d'intégrer du code à l'interface. Le lien entre le code et le *designer* n'est pas du tout intuitif, voire inexistant ; lorsqu'on ajoute un événement sur un composant, la méthode d'appel de l'événement est simplement générée à la fin de la classe ***StateMachine*.** Il n'est pas possible ensuite de le retrouver directement depuis le composant ; il faut le chercher manuellement dans tout le code de la classe ***StateMachine***. Dans la cas d'une application sérieuse, composée d'une dizaine de fenêtres, chacune munie de plusieurs composants, la taille de la classe **StateMachine au**gmente rapidement, conduisant rapidement à une situation difficilement gérable pour le développeur d'une part, source d'erreurs d'autre part.

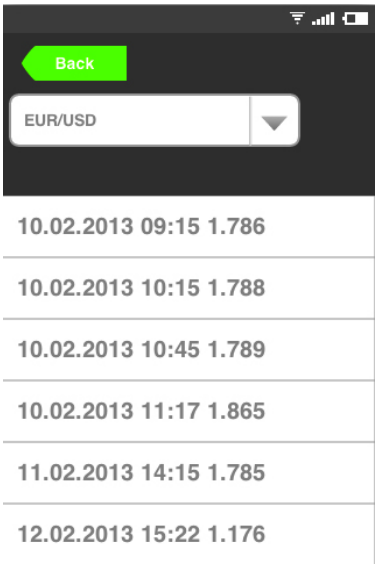
4. Mise en œuvre

Nous allons illustrer la mise en œuvre de *Codename One* en développant une application simple, composée de plusieurs écrans, qui aura recours à un service web et qui mémorisera des informations dans une base de données. Les caractéristiques de l'application modèle que nous allons réaliser ont été choisies ainsi car celles-ci se retrouvent dans de nombreuses applications proposées par les différents « *markets* ». Deux versions de l'application seront proposées, la première sera réalisée sans l'aide du *designer* d'interface et la seconde avec le *GUI Builder*.

4.1.Cahier des charges et prototype : Application de cours de change

L'application que nous allons réaliser sera constituée de trois écrans :

Le premier écran permet à l'utilisateur de choisir une conversion de devises ; une fois le choix effectué, l'application récupère le cours de change correspondant sur le site de *Yahoo Finance*. L'utilisateur peut choisir de sauvegarder ce cours afin de l'ajouter à un historique qui sera stocké dans une base de données ; il peut également effacer l'historique ou l'afficher. Ce dernier choix provoque l'affichage du deuxième écran. Enfin, cet écran affiche également le cours moyen des cours sauvegardés dans l'historique.



Le deuxième écran permet de consulter l'historique. Chaque cours est affiché avec la date et l'heure de son enregistrement. En sélectionnant un cours, l'utilisateur ouvre le troisième écran. Il est possible de revenir au premier écran.

Le troisième écran permet de calculer la conversion avec un montant choisi ; le cours appliqué est celui qui a été sélectionné dans le deuxième écran.

4.2.Compte rendu sur l'application finale

Nous allons maintenant décrire quelques parties du code, principalement la partie sur le *service web*, qui est un peu différente de la mise en œuvre par exemple sur *Android*. Vous trouverez en annexe de ce document les deux projets *NetBeans,* si vous souhaitez les consulter pour plus de détail.

Voici un aperçu de l'application finale, tournant sur i*OS*

Et la version *Android :*

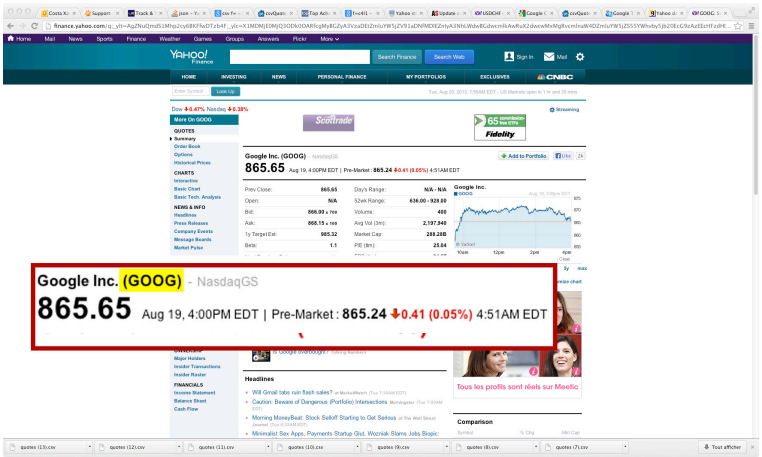
4.2.1. Service Web Yahoo

Yahoo finance propose un *web service* permettant de télécharger, sous forme de fichier «**.csv »,* les cours de la bourse en temps réel et gratuitement. Je me suis servis d'un tutorial trouvé sur *Google Code,* pour construire la requête permettant de récupérer le fichier *«*.csv ».*

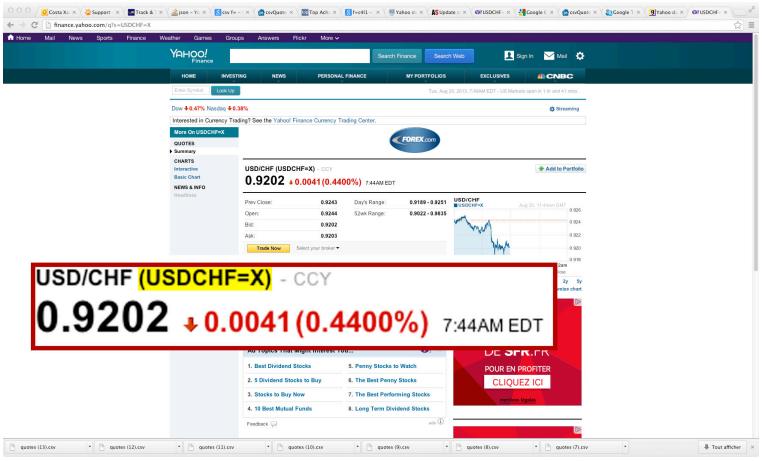
https://code.google.com/p/yahoo-finance-managed/wiki/csvQuotesDownload

Pour ce faire il suffit d'utiliser la requête « *http://finance.yahoo.com/d/quotes.csv?* » *et y rajouter les arguments nécessaires (*avec *« & »* entre chaque argument*):*

- *« s=* » Le cours que l'on veut récupérer. Il suffit d'aller sur le site *Yahoo Finance* pour récupérer le code du cours. Une fois le cours trouvé, via la barre de recherche, le code est affiché entre parenthèse à côté du cours (de l'action ou du change recherché(e)). Vous pouvez voir, sur l'image ci-dessous, que le code de l'action *Google* est *« GOOG ».*



Sur cette image, on a le code cours de change de *USD/CHF* qui est *« USDCHF=X ».* Les codes de cours de change ont le format *« nomChangeOrigineNomChangeDestination=X »*



Il est possible de récupérer plusieurs cours avec la même requête, en ajoutant une *« , »* entre chaque code de cours.

- *« f= »* les propriétés que l'on souhaite récupérer ; dans notre cas ce sera juste le dernier cours *: « l1 »* (attention à respecter la casse ici).
  Vous trouverez une liste des propriétés que l'on peut récupérer à cette adresse :
  https://code.google.com/p/yahoo-finance-managed/wiki/enumQuoteProperty

Ce qui donnera comme requête, par exemple, pour récupérer le cours *USD/CHF :*

http://finance.yahoo.com/d/quotes.csv?s=USDCHF=X&f=l1

Maintenant que l'on sait comment récupérer un cours, regardons comment on gère le *web service* dans *Codename One.* Ci-dessous, le code de la classe ***ServiceWebYahoo*** du projet réalisé, où est implanté ***ConnectionRequest***.

```
public class ServiceWebYahoo {

    String cours;

    public String getTaux(String change) {
        ConnectionRequest cmdReq = new ConnectionRequest(){
            protected void readResponse(InputStream in) throws IOException {
                InputStreamReader reader = new InputStreamReader(in);
                StringBuilder result = new StringBuilder();
                int car = reader.read();
                while (car != -1) {
                    result.append((char)car);
                    car = reader.read();
                }
                cours = result.toString();
            }
        };
        InfiniteProgress ip = new InfiniteProgress();
        Dialog dlg = ip.showInifiniteBlocking();
        cmdReq.setDisposeOnCompletion(dlg);
        cmdReq.setUrl("http://finance.yahoo.com/d/quotes.csv?e=.csv&f=c4ll&s="+change);
        cmdReq.setPost(false);
        NetworkManager.getInstance().addToQueueAndWait(cmdReq);
        return cours;
    }
}
```

Dans *Codename One*, il n'est pas possible d'utiliser les mêmes packages, que l'on utilise habituellement en Java, pour accéder à un *web service*. Une requête à un *web service* est modélisée par la classe ***com.codename1.io.ConnectionRequest***. Une instance de cette classe doit être insérée dans une file d'attente de requêtes gérée par la classe ***com.codename1.io.NetWorkManager***.

Ce qu'on peut remarquer comme différences de fonctionnement avec Java :

- Possibilité de traiter la réponse à la requête au *web service* de manière synchrone (ce que nous faisons ici en invoquant la méthode ***addToQueueAndWait***) ou de manière asynchrone (il faudrait alors appeler la méthode ***addToQueue*** à la place).
- La réponse est traitée dans la méthode ***readResponse*** **de** ***ConnectionRequest*** (qu'il faut donc redéfinir). Cette méthode sera déclenchée lorsque la requête au *web service* retourne son résultat.

4.2.2. Problèmes rencontrés

Lors du développement j'ai du faire face aux problèmes suivants :

- L'application fonctionne sur le simulateur de l'*IDE*, mais ne fonctionne pas sur le device (surtout sur *iOS*).
- La première version de mon application ne fonctionnait pas sur iOS, le problème venait de la partie base de donnée. Je testais si la table existe déjà, avant de faire une requête (« *SELECT », « INSERT », « DELETE »*). Si ce n'était pas le cas, la base était créée. Cela fonctionnait très bien sur *Android*, mais apparemment *iOS* ne le prenait pas en compte. J'ai donc remplacé ce test, en créant directement la table, lors de la création de la base de données afin de résoudre ce problème.
- Si l'on veut que l'application ressemble au mieux aux applications natives, on se voit obligé de créer un thème par plateforme. Par exemple, si vous utilisez le thème par défaut, vous n'aurez pas de différence entre un bouton « *enable »* et *« disable ».*
- La documentation des *packages* n'est pas assez détaillée. On n'a pas une explication complète de l'effet des méthodes.
- Le temps de compilation sur le *Cloud* est souvent très long, surtout pour *iOS*.

5. Bonnes pratiques

- Pour la gestion des exceptions, mettez des *boîtes de dialogue*, avec comme message le nom de la procédure source ainsi que l'exception générée. Il sera beaucoup plus facile ensuite, de trouver d'où proviennent les problèmes.
- Découpez votre application en plusieurs couches, afin de séparer les responsabilités. Si vous devez par exemple, modifier le *design* de votre application, vous n'aurez à modifier qu'une seule couche et pas le reste du code.
- Lorsque vous utilisez le *designer* de fenêtres, pensez à sauvegarder régulièrement. Il arrive souvent qu'il se bloque et que l'on doive le relancer ; tout le travail non sauvegardé est alors perdu.
- Effectuez régulièrement des tests sur chacun des *devices* cibles et pas seulement sur le simulateur de l'*IDE.* Le simulateur n'est pas une machine virtuelle de vos appareils mobiles ; l'application est lancée sur la machine virtuelle *JavaSE*[8] définie dans votre l'*IDE*. Vous ne pourrez pas voir les erreurs dues à la compatibilité de certaines fonctionnalités pour certaines plateformes.
- Si vous modifiez le nom de la classe ***Main*** (nom, emplacement, nom du package), le *« Refactor »* ne suffit pas. Il faut aller aussi dans les propriétés du projet et définir manuellement le nom de la fenêtre principale de l'application.
- Pensez à consulter la documentation en ligne afin de vérifier la compatibilité de certaines fonctionnalités avec les plateformes cibles de votre application. Il y est proposé parfois une alternative. Par exemple pour *SQLite,* dont la portabilité n'est pas assurée sur toutes les plateformes, il est conseillé d'utiliser la classe ***Storage*** à la place.

[8] **Java SE :** (anciennement Java 2 Standard Edition ou J2SE) est l'édition maitresse (ou distribution) du framework 'Plate-forme Java' d'Oracle, destinée typiquement aux applications pour poste de travail *Source Wikipedia* http://fr.wikipedia.org/wiki/Java_SE

6. Conclusion

Codename One, permet de développer rapidement une application multiplateforme, avec un seul langage de programmation. Pour une personne connaissant uniquement *Java*, *Codename One* est une bonne alternative afin de produire facilement une application pour plusieurs plateformes. Mais si votre but est d'avoir une application semblable en tout point à une application native et d'utiliser des composants spécifiques à chaque plateforme, alors *Codename One* ne sera pas la bonne solution.

La prise en main *du plug-in* est très rapide, si l'on a l'habitude de développer des applications Android. Il suffit de se familiariser avec les nouvelles librairies et le *designer* de fenêtres.

Le principal défaut de *Codename One*, de mon point de vue, est le fait que l'on n'a pas de vraies machines virtuelles de chaque plateforme mobile, en plus du simulateur. Ce qui nous oblige, si l'on veut être sûr que ça fonctionne sur la plateforme cible, de lancer une compilation sur le Cloud à chaque modification (ce qui peut prendre beaucoup de temps !). Cela peut être très gênant lorsque l'on doit débugger l'application.

Pour pouvoir comparer les limitations du *plug-in,* il aurait été intéressant de développer en parallèle l'application sur chacun des deux environnements de développement (*iOS et Android*). .

La sortie de *Codename One* date seulement d'un peu plus d'un an et cet environnement n'a rien à envier de ses concurrents. Son avenir est très prometteur si quelques modifications sont apportées dans le futur. Ma principale suggestion dans cette direction est de générer une ***StateMachine*** pour chaque fenêtre de l'application et de définir un processus simple permettant d'atteindre directement, depuis le GUI, le code généré pour les événements de chacun des composants. Enfin, la documentation des classes du *framework* pourrait être améliorée en explicitant mieux les descriptions des méthodes et en l'enrichissant d'exemples d'emploi.

Ce travail m'a beaucoup apporté sur le plan personnel, mais il m'a aussi permis d'améliorer mes compétences en programmation. Durant ma formation, à chaque nouvel outil ou programme que l'ont découvrait, nous avions quelqu'un pour nous

former. En effectuant ce travail, j'ai dû me former seul sur ce produit, ce qui n'a pas été forcément évident à certains moments ; mais les nombreuses erreurs que j'ai pu commettre m'ont permis de comprendre le fonctionnement de *Codename One*. La formation que j'ai suivie m'a quand même beaucoup aidé pour cela.

Lors du développement de l'application, j'ai rencontré quelques soucis de compatibilité avec iOS. C'est à ce moment que je me suis rendu compte que bien structurer son code et bien gérer les exceptions est important. En procédant ainsi, j'ai pu identifier les problèmes et les résoudre beaucoup plus rapidement.

7. Bibliographie

Sites internet consultés :

- Codename One Google Code. Javadoc Codename One [en ligne] https://codenameone.googlecode.com/svn/trunk/CodenameOne/javadoc/index.html (consulté le 20.08.2013)
- Google code.Yahoo ! Managed [en ligne] https://code.google.com/p/yahoo-finance-managed/ (consulté le 09.07.2013)
- Yahoo ! Finance. Cours de change [en ligne] http://finance.yahoo.com/ (consulté le 25.07.2013)
- Codename One. Discussion Forum [en ligne] http://www.codenameone.com/discussion-forum.html (consulté le 20.08.2013)
- Codename One [en ligne] http://www.codenameone.com/index.html (consulté le 12.08.2013)
- Appcelerator. Titanium platform [en ligne] http://www.appcelerator.com/platform/titanium-platform/ (consulté le 14.08.2013)
- PhoneGap. PhoneGap Documentation [en ligne] http://docs.phonegap.com/en/3.0.0/index.html (consulté le 14.08.2013)
- Retroweaver. FAQ [en ligne] http://retroweaver.sourceforge.net/ (consulté le 17.08.2013)
- XMLVM. Overview [en ligne] http://xmlvm.org/ (consulté le 17.08.2013)
- Codename One. Developper Guide [en ligne] http://www.codenameone.com/developer-guide.html (consulté le 25.06.2013)