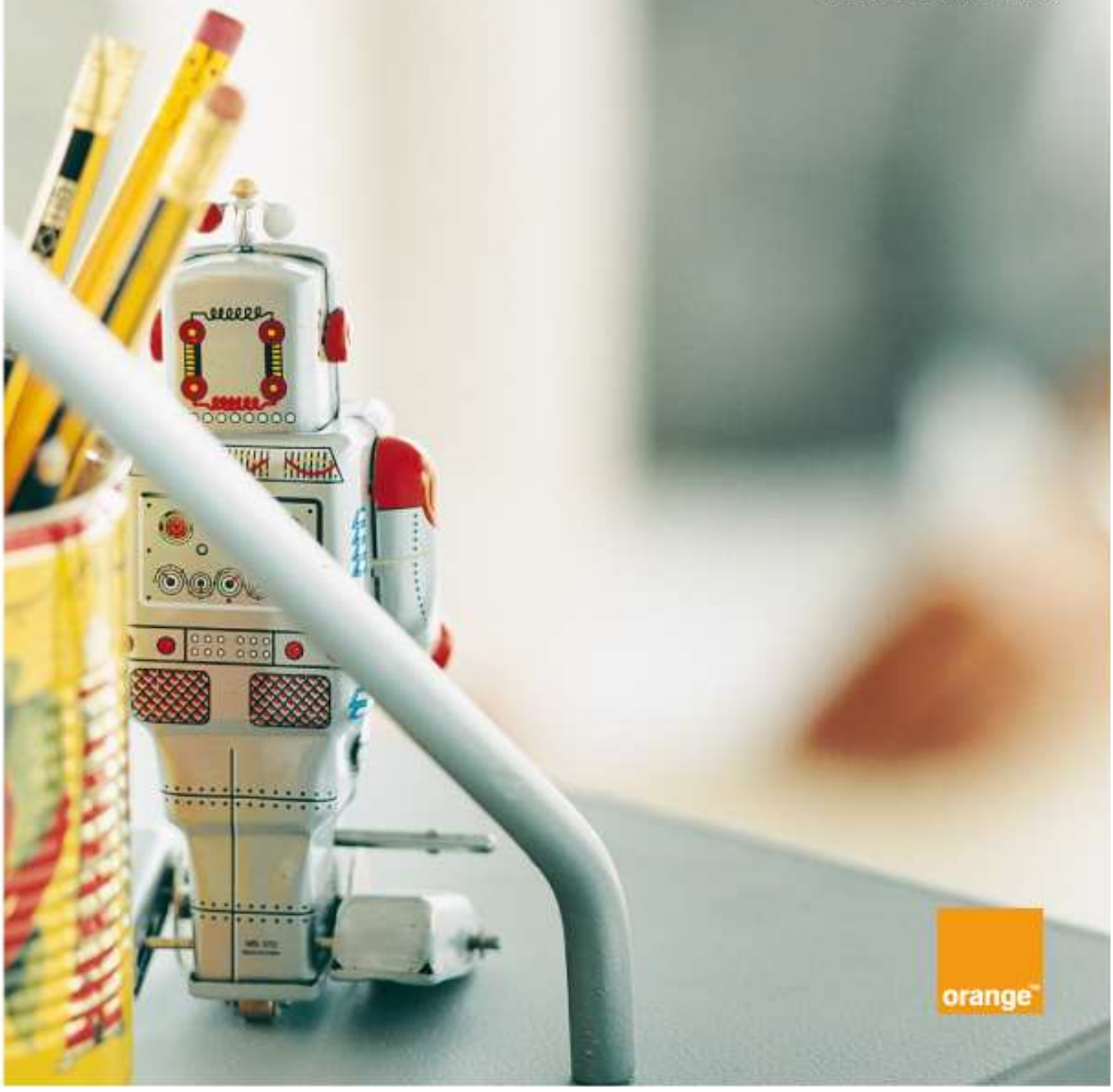
user guide

iPhone & ANDROID

édition Octobre 2010



orange

Guide pratique de développement pour appareils mobiles

Première édition

Introduction

Depuis son lancement le 5 mai 2010, Orange est devenu un acteur incontournable des télécommunications en Tunisie.
Le lancement de services innovants et inédits tels que l'Iphone 4, la Flybox ou encore les clés 3G+ a permis de susciter le développement de nouveaux usages et peut créer de nouveaux marchés.

Entreprise citoyenne, Orange souhaite utiliser sa forte capacité d'innovation pour encourager la fibre créatrice des développeurs d'applications mobiles, relais de croissance puissants pour la Tunisie.

Pour cela, Orange met à la disposition des développeurs un Centre d'Expertise dédié aux technologies mobiles, en particulier iPhone et Android.

Le Centre propose un accompagnement allant de la formation à la validation des applications en passant par une assistance à la commercialisation sur la marché tunisien et international.

Ce guide est une donc première étape pour vous accompagner vers vos futurs projets mobiles.

En quelques mots

Ce guide pratique est autant destiné aux développeurs novices qu'aux personnes étant amenées à gérer un projet de développement mobile.

Il présente dans un premier temps les différents environnements applicatifs avec leurs avantages et leurs inconvénients. Puis il détaille les rudiments du développement pour iPhone et Android, les deux OS actuellement les plus populaires auprès des acheteurs et des développeurs mobiles.

En lisant ce guide, vous découvrirez les frameworks de développement et les outils associés afin de commencer à développer, tester, et distribuer votre programme pour ces OS mobiles.

Développer une application mobile est relativement simple. Il faut néanmoins disposer de connaissances en programmation orientée objet et être disposé à se pencher sur les spécificités des langages, des OS et des terminaux.

Avec un peu de pratique et de recherches sur Internet, le monde fascinant des applications mobiles est à la portée de tout développeur motivé.

À vos claviers !

Environnements applicatifs

Il existe plusieurs formes d'applications mobiles. Les plus communément utilisées sont les applications natives et les sites Internet.

Applications natives

De nos jours, il existe de nombreux systèmes d'exploitation (OS) mobiles sur le marché, dont certains sont open source. Les plus importants sont Android, Bada, Blackberry, iOS, Meego, Symbian, webOS et Windows Mobile. Il est possible de créer des applications natives pour chacun de ces OS sans demander d'autorisation à qui que ce soit.

Les applications natives sont appréciées pour leur meilleure intégration au système et leurs excellentes performances. Néanmoins, elles demandent un important effort de développement et doivent être réécrites pour chaque plate-forme.

De plus, la plupart des téléphones grand public contiennent des systèmes d'exploitation qui ne permettent pas d'inclure du code natif créé par un tiers comme par exemple les gammes *Nokia Series 40, Samsung SGH* ou encore *Sony Ericsson Java Platform.*

Le tableau suivant présente rapidement les différents systèmes d'exploitation mobiles du marché :

Android et iOS sont les deux OS de smartphones se vendant le mieux au monde. Symbian dispose de la plus grande base installée mais perd des parts de marché mois après mois.

Système d'exploitation	Langage(s) De programmation	Remarques
Android	Java, C	OS open source (dérivé de Linux) developer.android.com
iOS (iPhone)	Objective-C, C	OS propriétaire. Requiert un compte de développeur Apple developer.apple.com/iPhone
Symbian	C, C++, autres	OS open source en parte de vitesse developer.symbian.org
Bada	C++	OS propriétaire disponible uniquement sur Samsung developer.bada.com
Blackberry	Java	Également compatible J2ME, mais les API natives fournissent une meilleure intégration na.blackberry.com/eng/developers
Meego	C, C++	OS open source (Linux). Fusion de Maemo et Moblin meego.com/developers
webOS	HTML, CSS, JavaScript	Programmation de type widget, OS propriétaire dérivé de Linux developer.palm.com
Windows Mobile	C#, C	Programmation directement sur Windows Mobile ou via .NET Compact Framework. J2ME généralement disponible grâce à une JVM pré-installée developer.windowsphone.com

Sites Internet

Les pages Web sont compatibles avec tous les smartphones et devraient être en principe l'environnement de développement de choix pour une compatibilité maximale.

Néanmoins le nombre de navigateurs et leurs différences de fonctionnalités posent problème. En effet, si

certains navigateurs sont très puissants et supportent le CSS autant que le JavaScript, d'autres ne gèrent que le XHTML, ce qui limite considérablement leur potentiel. Heureusement, les vieux standards WAP avec ses pages WML ne sont plus d'actualité.

Par ailleurs, les applications construites à partir de pages Web sont moins performantes et n'accèdent qu'à une partie des fonctionnalités des téléphones. Elles restent néanmoins simples à développer et leur déploiement ne nécessite pas d'intermédiaire.

Développer des applications iPhone

Présentation

L'iPhone est une plate-forme très intéressante et très populaire pour de nombreuses raisons, dont la principale est sans aucun doute l'App Store. Lors de son apparition en juillet 2008, un an après le lancement du premier iPhone, l'App Store s'est imposée comme une place de marché incontournable. Depuis son lancement, plus de 6 milliards de téléchargements d'applications ont été effectués par ses utilisateurs. À ce jour, plus de 300 000 applications sont disponibles. Ce chiffre démontre le succès incontestable du concept mais également alerte les développeurs qu'il est laborieux d'être visible dans cette masse d'applications.

Les APIs de haute qualité fournies avec le SDK de l'iPhone couvrent un nombre important de tâches permettant de réduire sensiblement le temps de développement d'une application.

Plus de 1 500 APIs ont été ajoutées à la dernière mise à jour majeure d'iPhone OS 4 (ou iOS 4).

Pré-requis au développement sur iPhone

Pour commencer à développer sur iPhone (et iPod Touch), vous avez besoin du SDK qui peut être téléchargé sur http://developer.apple.com/devcenter/ios/index.action.

L'inscription est obligatoire et gratuite (http://developer.apple.com/programs/register/). Si vous souhaitez tester votre application sur votre iPhone (iPod ou iPad) et la distribuer sur l'App Store, une souscription annuelle de $99 est nécessaire.

Le iOS SDK contient plusieurs outils qui vous permettront de programmer, de tester, et de débugger vos applications. Les outils les plus utilisés sont :

- Xcode, l'IDE pour le SDK de l'iPhone
- Interface Builder, pour construire des interfaces utilisateur pour des applications iPhone
- Instruments, qui offrent différents outils pour contrôler l'exécution des applications
- iPhone simulator, qui permet au développeur de tester une application plus rapidement que sur un iPhone réel.

Le SDK de l'iPhone fonctionne sur un processeur Intel, avec Mac OS X 10.5 (Leopard) ou 10.6 (Snow Leopard).

Des références et des guides sont disponibles en ligne sur developer.apple.com/library/ios/navigation.

Implémentation

D'une manière générale, vous préférerez utiliser Cocoa, l'API de gestion du multitouch, impliquant que vous programmerez en Objective-C et devrez créer vos UI dans Interface Builder, qui utilise le format de fichier propriétaire XIB.

Objective-C est un langage de programmation orienté objet basé sur C. Il est pleinement compatible avec C, ce qui signifie que vous pourrez utiliser votre code C directement dans votre fichier Objective-C.

Si vous développez sur un autre langage orienté objet comme C++ ou Java, programmer en Objective-C nécessitera un apprentissage complémentaire. Le centre d'expertise de Orange Tunisie et ses experts sont à votre disposition sur rendez vous pour vous accompagner à débuter sur ce langage de programmation.

Pour commencer, nous vous conseillons le « Guide Apple pour créer sa première application iPhone »[1] (en anglais) qui explique différents concepts et tâches indispensables dans le workflow de production.

Vous trouverez également des exemples de code[2] qu'Apple fournit en ligne afin d'appréhender les différentes API qui sont disponibles.

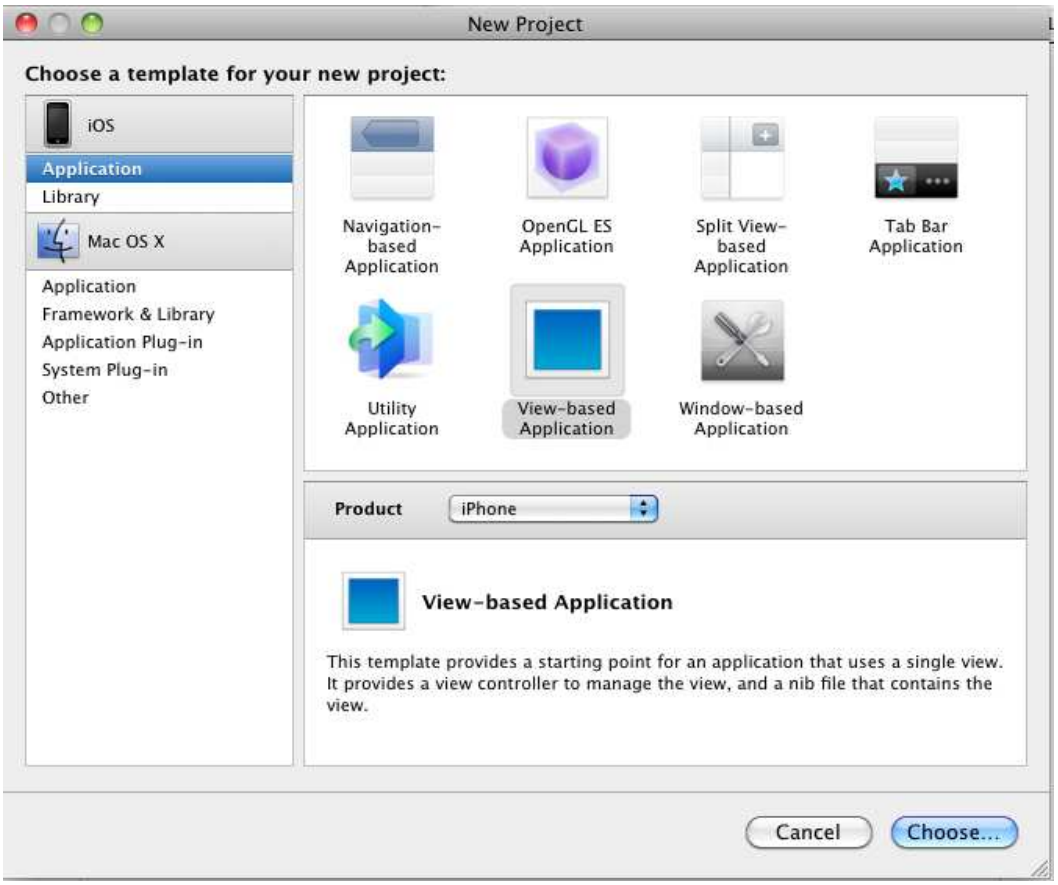
[1] http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/Creating_an_iPhone_App/index.html%23//apple_ref/doc/uid/TP40007595

[2] http://developer.apple.com/library/ios/navigation/index.html

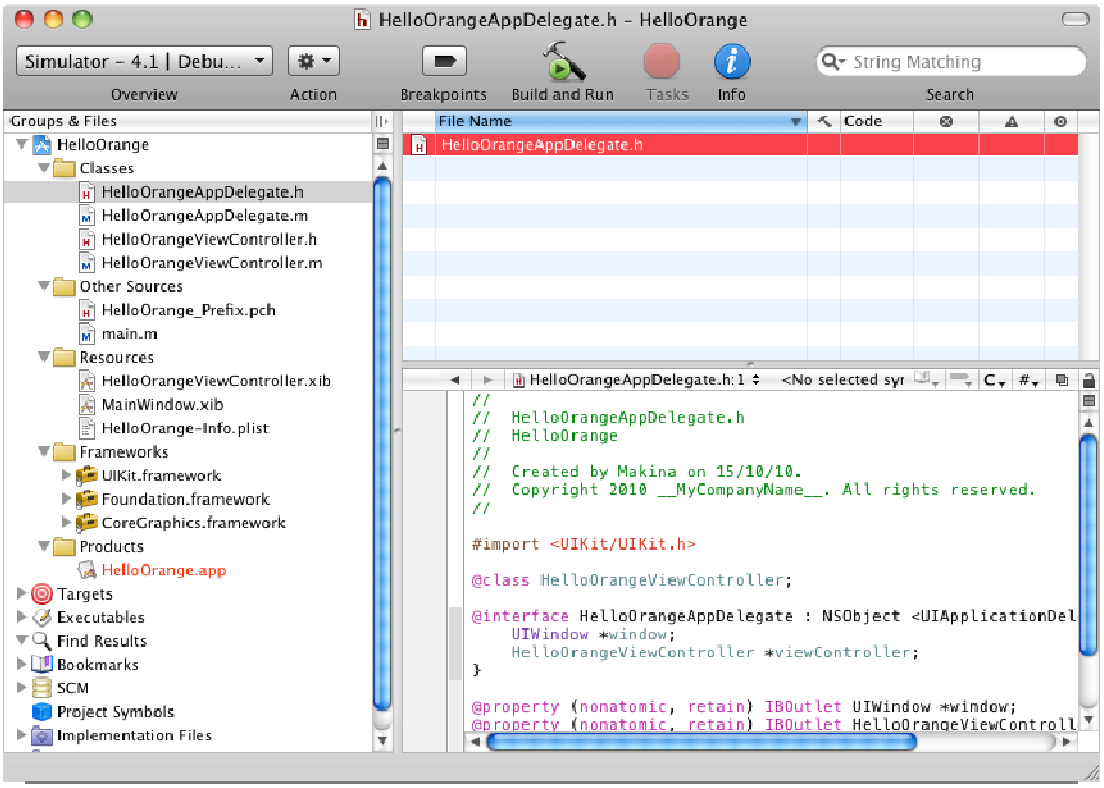
Création d'une première application

Nous vous proposons de créer votre première application iPhone avec XCode en commençant avec l'application « HelloOrange ».

L'application « HelloOrange »



Dans un premier temps, lancez l'application XCode, puis sélectionner la création d'un nouveau projet « File > New Project… » de type « View-based application ». Valider votre choix en double cliquant .



Nommez-le projet « HelloOrange » et sauvegardez le dans le répertoire de travail de votre choix. Lors de la création de votre projet, Xcode génère automatiquement des fichiers avec des lignes de code pour que votre application iPhone puisse s'exécuter correctement.

Déployez les répertoires « Classes », « Other Sources », « Resources » et concentrez vous sur les 3 fichiers suivants :

1. Le fichier **helloOrangeViewController.h** permet de définir toutes les propriétés et actions qui permettront d'afficher le texte « Hello Orange » sur l'écran de votre simulateur iPhone.

2. Le fichier **helloOrangeViewController.m** dans lequel vous implémenterez le code des méthodes et actions prédéfinies dans le fichier header.

Retenez que le .h et le .m marchent toujours par paire.

3. Le fichier **helloOrangeViewController.xib** qui est l'interface utilisateur qui exécuera le programme Interface Builder afin de créer les interfaces graphiques de votre application.

Pour commencer, ouvrez le fichier **helloOrangeViewController.h** en cliquant dessus. Dans votre application vous aurez besoin d'un label. C'est dans ce label que vous afficherez votre texte « Hello Orange », ainsi qu'un bouton qui déclenchera l'affichage de votre texte.

Vous allez donc définir les propriétés entre des accolades. Il vous faudra donc un UILabel.

```
#import <UIKit/UIKit.h>

@interface HelloOrangeViewController : UIViewController {

    UILabel *myLabel;
    UIButton *myButton;
```

```
}

@property (nonatomic, retain) IBOutlet UILabel *myLabel;
@property (nonatomic, retain) IBOutlet UIButton *myButton;

- (IBAction)buttonPressed:(id)sender;

@end
```

Ensuite dirigez vous vers le fichier d'implémentation (.m) : **helloOrangeViewController.m**

```
#import "HelloOrangeViewController.h"

@implementation HelloOrangeViewController

@synthesize myLabel;
@synthesize myButton;

- (IBAction)buttonPressed:(id)sender
{
    [myLabel setText:@"Hello Orange"];
}
```

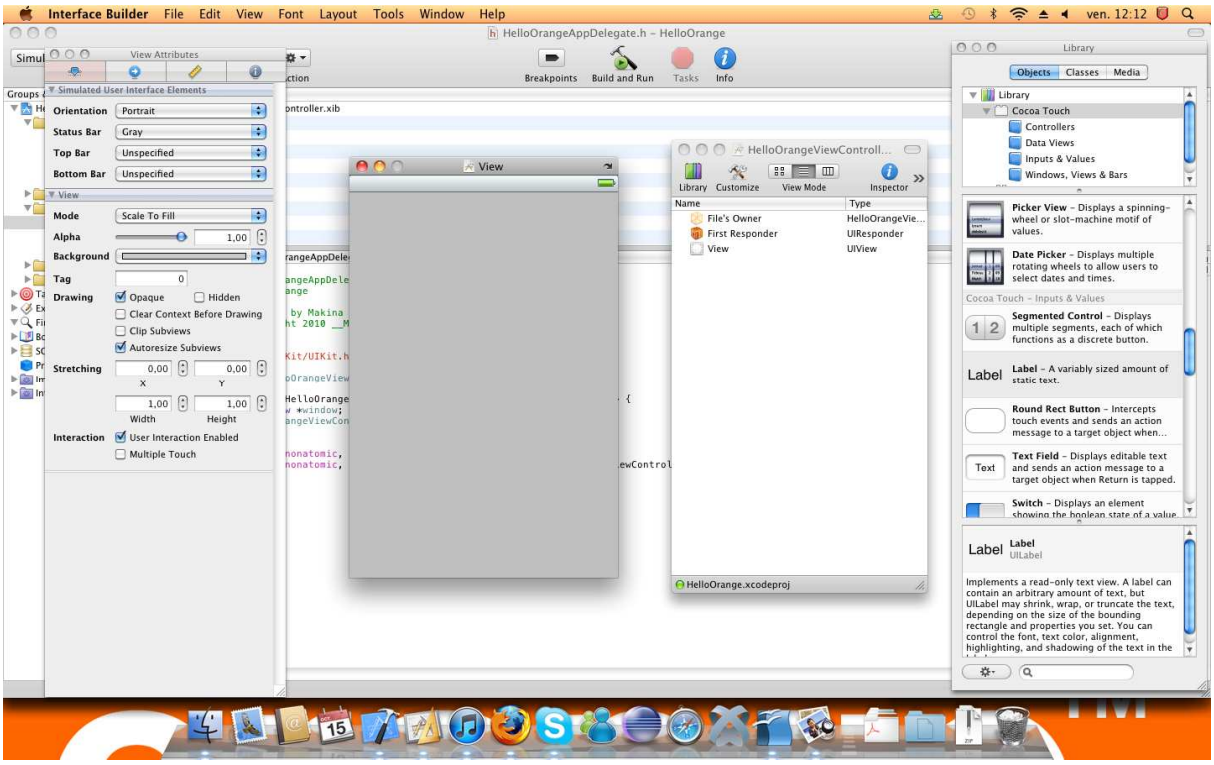
Dans ce fichier vous n'avez pas besoin de re-déclarer les propriétés, vous ne déclarerez que les méthodes dont vous vous servirez.

Utiliser Interface Builder pour « dessiner » votre application :

Votre code qui permet d'afficher votre texte « Hello Orange » sur l'écran de votre simulateur iPhone est prêt. C ependant il vous manque un élément clé qui est l'Interface Utilisateur.

Dans le répertoire « Resources » cliquez deux fois sur le fichier **helloOrangeViewController.xib**.et 4 fenêtres vont s'ouvrir :

1. **Library** (ou Bibliothèque en Français) qui permet de sélectionner vos éléments graphiques tels que UILabel, UITextField, UIButton, etc.
2. **View** qui est en quelque sorte votre écran iPhone. Vous glisserez vos éléments graphiques ici et les organiserez de manière à facilité l'usage de votre application.
3. **helloOrangeViewController.xib** qui affiche en fonction du mode de visualisation les éléments présents dans le fichier .xib
4. **Inspector** (pour l'afficher appuyer sur ⌘+1, ou Tools > Inspector) qui regroupe 4 autres fenêtres segmentées par des onglets qui vous permettent de modifier les propriétés de vos objets UILabel, View, etc.

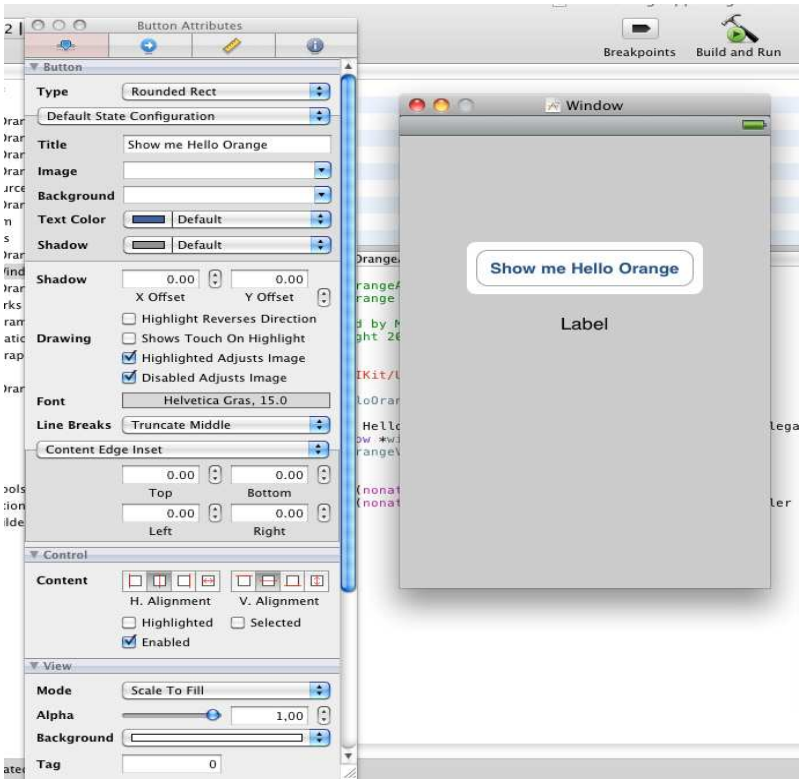


Vérifier que vous êtes dans l'onglet « object » dans la fenêtre « Library » et dans le moteur de recherche indiqué en bas avec une loupe, saisissez UILabel jusqu'à ce que la bibliothèque (ou « Library ») « Label » s'affiche dans la liste au-dessus.

Ensuite il ne vous restera plus qu'à simplement glisser-déposer votre bibliothèque « Label » dans la fenêtre « View ».

Dans la fenêtre « **helloOrangeViewController.xib »** sélectionnez l'affichage en mode vue et déployez le répertoire « View ». Vous constaterez que votre blibliothèque « Label » y a été ajoutée.

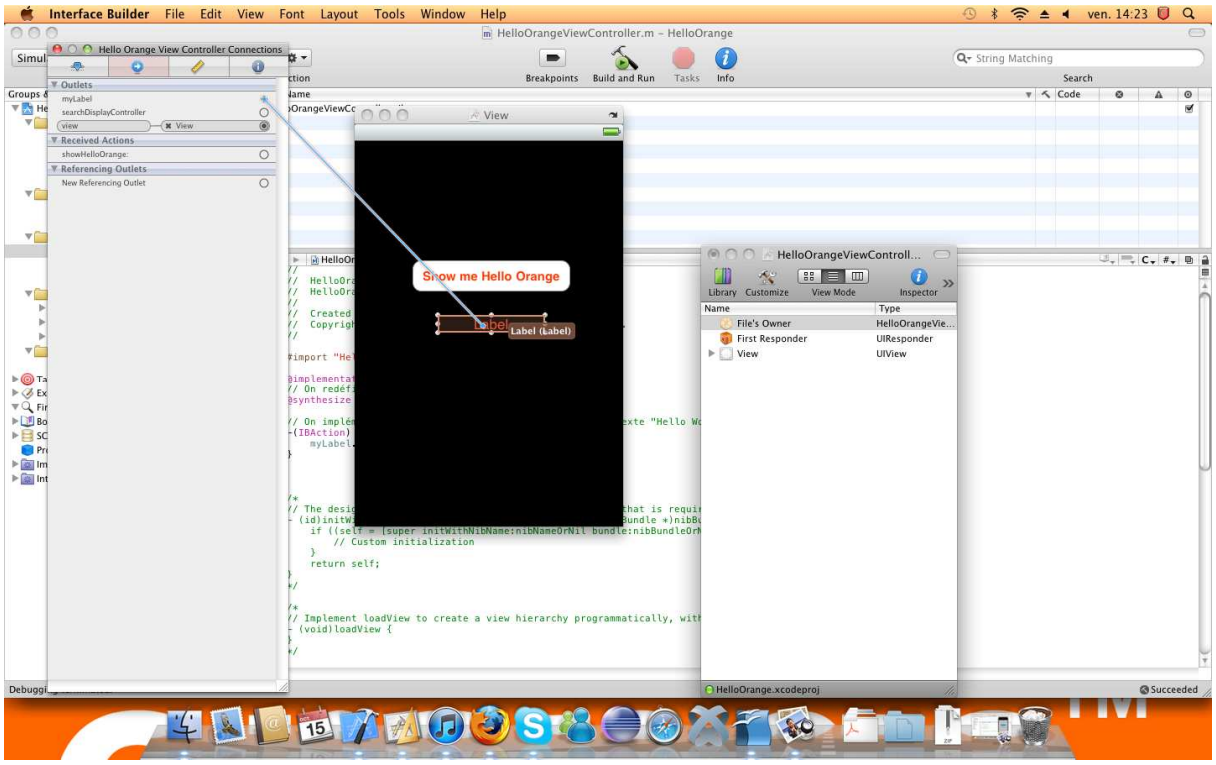
Répétez l'opération pour UIButton et ajouter dans la fenêtre « View » la bibliothèque « Round Rect Button ». Sélectionnez le bouton et redimensionnez-le à la taille de votre convenance.



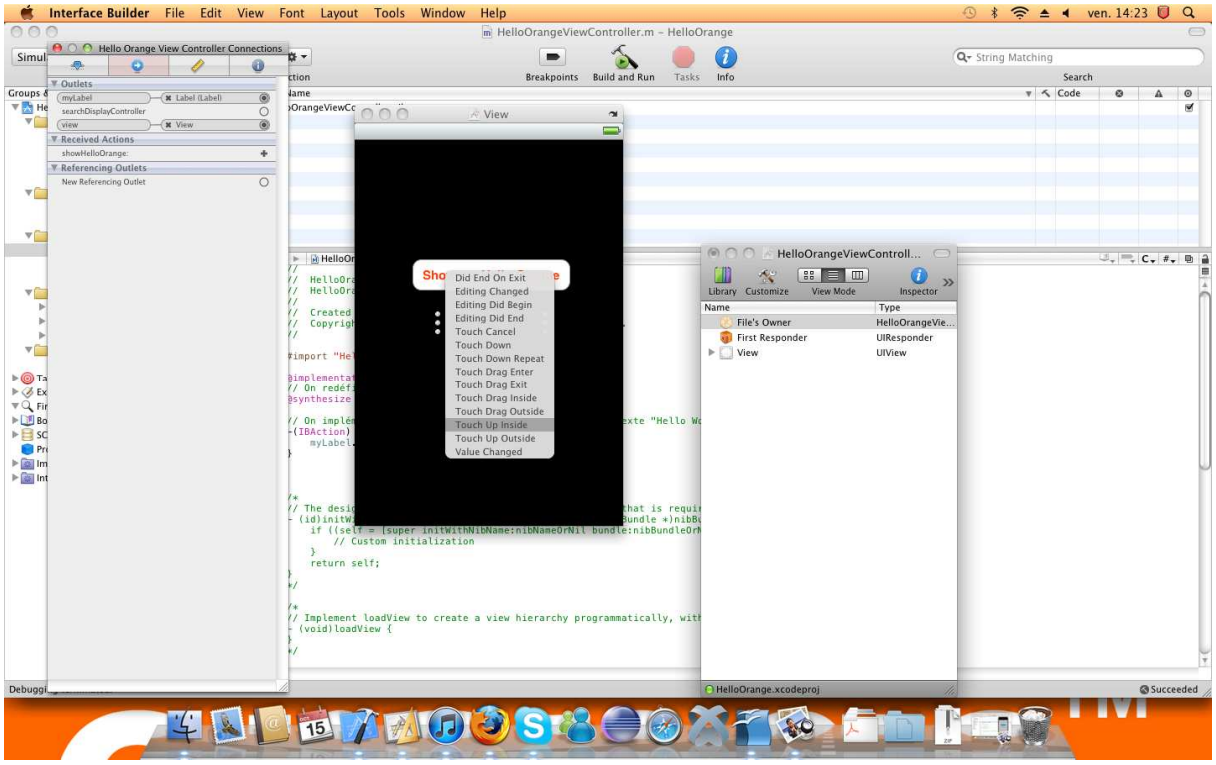
Cliquez à l'intérieur du bouton, sélectionnez la vue « Attributs » (premier onglet à gauche) dans « Inspector », et dans le titre (*Title*) écrivez par exemple « Show me Hello Orange »

Vous allez maintenant lier vos éléments graphiques à votre code.

Pour cela, déplacez vous sur le nom « File's Owner » dans la fenêtre « **helloOrangeViewController.xib** », effectuez la combinaison du bouton gauche de votre souris avec la touche enfoncée [ctrl] de votre ordinateur Apple Mac et sans lâcher votre clic déplacez-vous sur votre « Label ». Une popup Outlets s'affichera alors. Vous pourrez ainsi choisir myLabel. Réalisez la même opération en vous déplaçant sur le bouton et choisir myButton. Vous venez donc de lier votre code à votre interface graphique.



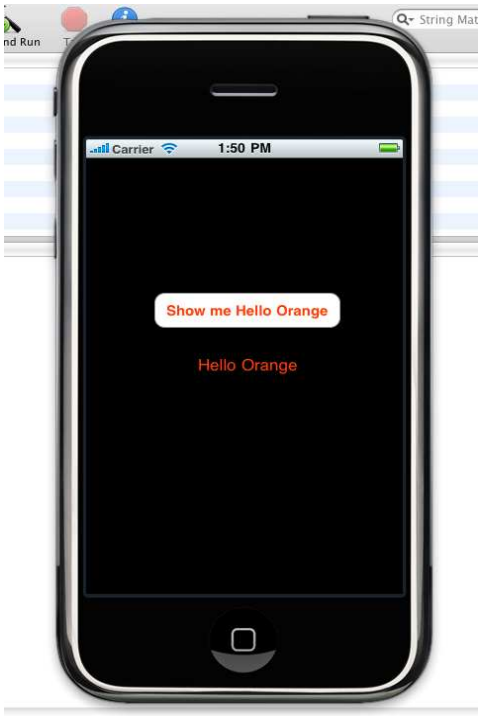
Maintenant il est nécessaire de lier votre action à votre méthode showHelloOrange.



Pour cela, sélectionnez votre bouton dans lequel vous avez saisi précédemment le message « Show me Hello Orange », dans Inspector, choisissez l'onglet « Connections » indiqué par une flèche. et une liste de toutes les actions possibles sur ce bouton s'affiche.

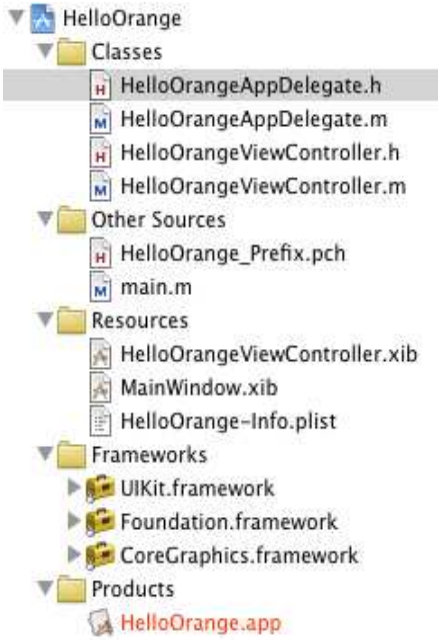
Vous allez vous intéresser à l'action « Touch Up Inside » qui correspond à la touche sur le bouton. Cliquez enfoncé sur le cercle a coté et déposez sur « File's Owner » et une popup apparaîtra, puis sélectionnez showHelloOrange. Vous venez donc de lier votre action à votre méthode, qui sera exécutée lorsque vous cliquerez sur votre bouton.

Enfin, pour tester votre première application, cliquez dans XCode le bouton « Build and Run » pour obtenir le résultat suivant:



Toutes nos félicitations, vous venez de créer votre première application pour iPhone.

Structure d'un projet iPhone



Un projet iPhone se compose de cinq sections :

- La section « Classes » contient toutes les classes de votre projet.
- La section « Other Sources » contient les autres fichiers sources qui ne sont pas des classes. Entre autres le "main.m" que vous ne devez pas toucher.
- La section « Ressources » contient des fichiers .xib. Ces fichiers sont des objets pré-construits à l'aide de l'application **InterfaceBuilder**.

Il y a aussi un fichier nommé **info.plist**. Le format **plist** (pour *Property List*) est un format XML spécial qui permet d'avoir une correspondance directe entre les fichiers. Ce fichier contient des informations de configurations de votre application.

- La section « Frameworks » contient les "includes" des Frameworks utilisés dans le projet. Par défaut, vous avez les 3 principaux: UIKit, Foundation et CoreGraphics.
- La section « Product » contient le fichier généré par la compilation, dans notre cas : HelloOrange.app

Tester

Le simulateur iPhone (et iPad) est utile pour tester son application en cours de développement. Toutefois il est indispensable de valider votre application dans un environnement client réel, à savoir sur un mobile iPhone 4.

Il est possible de faire tester votre application par 100 beta-testeurs de votre entreprise ou établissement scolaire via le portail de développeur d'Apple pour lequel vous êtes inscrit.

Chaque iPhone (iPod Touch ou iPard) détient un numéro d'identification unique (UDID, *universal device identifier*), qui est une chaîne unique de 40 caractères hexadécimaux basée sur différentes parties du hardware de l'appareil.

Distribution

Pour cibler votre audience au maximum, nous vous invitons à distribuer votre application sur l'App Store de Apple.

Pour valider la soumission de votre application sur l'App Store, vous aurez besoin de fournir à Apple depuis leur portail iTunes Connect :

- un icône au format 512x512 pixels,
- 5 captures d'écrans,

- le programme de votre application (ou « build ») correctement signée par un certificat de distribution.

Après la validation d'Apple, qui en moyenne n'excède pas deux semaines, votre application est immédiatement disponible sur l'App Store.

Il est nécessaire de respecter la charte de Apple pour éviter le rejet de votre application.

Pour cela, nous vous invitons à prendre connaissance des lignes directrices sur ce site Internet : https://developer.apple.com/appstore/resources/approval/guidelines.html

Communauté

La communauté des développeurs iPhone est très active et très riche en terme de contenu. De nombreux sites communautaires sont disponibles, parmi lesquels :

- Le site officiel d'Apple (EN) :http://developer.apple.com/devcenter/ios
- Le site d'entraide entre développeurs Stackoverflow (EN) : http://stackoverflow.com/questions/tagged/iPhone

à retenir

iPhone	Très bon	Les plus
		• Cohérence graphique et standardisation du hardware • Simplicité d'élaboration d'UI
Simplicité de développement	+ + + +	Les moins
Cohérence de l'UI	+ + + + +	• Dépendance vis à vis d'Apple pour la validation et la distribution d'applications
Simplicité de distribution	+ + +	• Développement sous Mac OS X seulement
Utilisateurs potentiels	+ + + + +	• Pas de *garbage collector* pour la gestion de mémoire

Développer des applications Android

Android est un système d'exploitation dérivé de Linux et accompagné d'un environnement de développement comportant tous les outils nécessaires à la création d'applications.

Présentation

Créé par Google et l'Open Handset Alliance fin 2007, la plate-forme Android est une des plus jeunes et la plus dynamique du marché. La version 1.0 apparaît durant l'été 2008, en même temps que l'*Android Market*. La version 2.0 date d'octobre 2009, la version 2.2 de juin 2010. Plus de 100 000 applications ont été recensées à ce jour pour cette plate-forme. Android a en outre été porté et est installé sur certains netbooks et tablettes. L'année 2011 verra le lancement de Google TV, fonctionnant également sous Android.

Ce système est installé sur les téléphones de nombreux constructeurs tels que HTC, Samsung, LG ou Sony Ericsson. En 2010, pas moins de cinquante nouveaux smartphones Android sont attendus : les constructeurs et les opérateurs peuvent installer l'OS gratuitement et le personnaliser très finement : interface, applications par défaut, services associés.

De nombreuses applications sont également pré-installées, la plupart mettant en avant les services de Google comme par exemple Google search, Google maps ou Gmail.

Caractéristiques des applications Android

Une application Android mêle activités et services. Une activité est une micro-fonctionnalité combinée à une interface utilisateur. Un service est une tâche exécutée en fond et ne correspond à aucune représentation visuelle. La communication entre activités, services et les autres parties du système comme la gestion du réseau est assurée par des « intentions ». Une intention décrit une action qui doit être réalisée. Elle contient des informations comme la position du toucher de l'utilisateur sur l'écran avec une signification sémantique. Par exemple, l'intention de montrer une page Web ouvrira le navigateur Web.
Une application sera toujours signée, grâce à une signature de débogage ou une signature définitive. Celle-ci pouvant être auto-signée, oubliez donc les frais de signature !

Développer

Le principal langage de programmation pour Android est Java 5. Néanmoins, la totalité de la bibliothèque Java n'est pas supportée et il existe un grand nombre d'API spécifiques.
Pour commencer, le SDK d'Android est nécessaire ainsi qu'un IDE Java. Le SDK est disponible pour Windows, Mac OS X et Linux et contient les outils pour construire, débugger et analyser des applications. Il n'y a donc pas de contrainte pour votre plate-forme de développement.
Concernant l'IDE, vous pouvez installer le plugin ADT pour Eclipse qui gère bien le développement et le déploiement. Mais comme le cycle de développement d'Android est basé sur Ant, vous pouvez utiliser l'IDE qui vous convient le mieux. Pour vous aider, vous disposez également de nombreux outils :

- android : créer un projet initial ou gérer des appareils émulés
- adb : Android Debug Bridge : permet l'accès à la console du téléphone connecté sur port USB ou à l'émulateur pour effectuer différentes tâches comme déplacer des fichiers, modifier les coordonnées GPS...
- émulateur : simule un appareil virtuel en émulant ses fonctionnalités. Lent à démarrer, il doit être lancé une fois et non à chaque build.
- ddms : analyse un appareil ou un émulateur en suivant les messages de log et en contrôlant ses fonctionnalités comme la latence réseau ou la position GPS. Contrôle la consommation de mémoire ou permet de fermer un processus. Lorsque cet outil est en fonctionnement, le débogueur de processus Eclipse peut également être connecté à un processus tournant dans l'émulateur.
- traceview : analyse les logs spéciaux qui sont activables via DebugstartMethodTracing ("myApp"); et permet de voir quelle méthode a fait quoi à quel moment.

À noter la nouvelle application Web App inventor (appinventor.googlelabs.com) de Google qui permet un développement simplifié via une interface graphique. Encore en bêta, ce service est accessible sur invitation.

Création d'une première application

Avant de commencer, il faut vous assurez que vous avez déjà installé le SDK Android ainsi que le plugin ADT pour Eclipse. Si tel n'est pas le cas, de nombreuses ressources en ligne peuvent vous aider : http://www.androideur.com/installer-android-sdk-sous-windows-part1 pour la préparation de l'environnement de développement Android puis installer Android SDK sous Windows en suivant ce tutoriel http://www.androideur.com/installation-d-android-sdk-sous-windows-deploiement

Sous Linux, vous pouvez notamment suivre ces indications : http://wiki.smartphonefrance.info/sdk-android-linux.ashx

Installer la plate-forme Android

Tout d'abord il faut installer au moins une des versions de l'OS Android . Si vous hésitez sur le choix de la plate-forme, sélectionnez la dernière mise à jour. Le choix de la plate-forme sera décisif en terme de compatibilité de l'application avec les différents appareils du marché.

Ainsi, assurez vous de téléchargez la plate-forme correspondante à la version disponible sur votre smartphone.

Créer l'émulateur AVD

Il est nécessaire d'utiliser un émulateur pour tester les applications et visualiser le comportement de l'interface et du système. L'émulateur du SDK Android s'appelle un AVD : Android Virtual Device.

Création de l'AVD dans Eclipse :

Menu **Window > Android SDK and AVD Manager**

- Choisir **Virtual Device**
- Choisir **New pour faire apparaître** le dialogue **Create New AVD**
- Choisir un nom pour l'AVD, **« premier_avd »** par exemple
- Choisir la plate-forme cible (« target ») correspondant à la version de l'image Android choisie
- Choisir **Create AVD**

Il est également possible de créer un AVD en mode console.

Créer un nouveau projet Android

Après avoir créé un émulateur de terminal, il faut créer un nouveau projet Android dans Eclipse :

- Menu **File > New > Project**
- Dans le dialogue Choisir **Android > Android Project > Next**
- Fournir des informations sur le projet :
  - **Project name** : HelloOrange
  - **Application name**: Hello, Orange
  - **Package name**: com.Orange.helloorange (ou votre propre namespace)
  - **Create Activity**: HelloOrange
  - **Build Target** : Android 2.2 (ou la version choisie)
- Choisir **Finish**

Signification des différents champs

- **Project name** : nom du répertoire contenant tous les fichiers du projet.
- **Application Name** : nom de l'application (apparaissant dans le menu du mobile)
- **Package Name** : namespace du package (suivant les mêmes règles qu'en Java) dans lequel réside le code du projet. Le nom de chaque package doit être unique dans chaque système Android. Il est donc

conseillé d'utiliser un nommage sous forme de domaine (com.Orange par exemple).

- **Create Activity** : nom du class stub créé par le plugin ADT. C'est une sous-classe de la classe Activity d'Android.
- **Min SDK Version** : définit l'API level minimum (version d'Android) requis par l'application (voir http://developer.android.com/guide/appendix/api-levels.html).
- **Use default location** permet de changer le répertoire où les fichiers du projet sont générés.

Build Target : précise la version de la plate-forme pour laquelle l'application sera compilée. Par défaut cette valeur est fixée par le paramètre MinSDK.Ce premier projet est désormais visible dans le Package Explorer. Le fichier HelloOrange.java situé dans **HelloOrange >src >com.Orange.helloorange** contient :

```
package com.Orange.helloorange;

import android.app.Activity;
import android.os.Bundle;


public class HelloOrange extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
       setContentView (R.layout.main);
    }
}
```

Cette classe est construite sur la classe *Activity* . Une Activity est une partie de l'application gérant un cas d'utilisation (des actions). Une application peut avoir plusieurs Activities séparées, mais l'utilisateur ne peut interagir qu'avec une seul d'entre elles à la fois.

La méthode *onCreate()* sera appelée par le système Android lors du démarrage de l'*Activity* . Il faut donc l'utiliser pour lancer la configuration de l'initialisation et de l'interface utilisateur. Une Activity offre généralement une interface d'interaction avec l'utilisateur même si ce n'est pas obligatoire.

Création de l'Interface Utilisateur (UI)

Modifiez le code pour changer le comportement :

```
package com.Orange.helloorange;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloOrange extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Orange");
        setContentView(tv);
```

```
        }
    }
```

Une Interface Utilisateur Android est composée de hiérarchies d'objets appelé *Views* (Vue). Une *View* est un objet se dessinant pour constituer une élément de l'interface utilisateur comme un bouton, une image ou du texte comme dans le cas présent. Chacun de ces objets est une sous-classe de la classe *View* ; la sous-classe gérant le texte est *TextView*.

Le code ci-dessus crée un *TextView* avec le constructeur de classe acceptant comme paramètre une instance d'un *Context* Android. Un *Context* fournit des services comme l'accès aux ressources, à une base de données, aux préférences… La classe *Activity* hérite du *Context* et comme la classe *HelloOrange* est une sous classe d'*Activity*, elle est aussi un *Context*. C'est ainsi que « this » peut être passé comme référence du *TextView*.

Le contenu textuel se définit avec *setText().*

Enfin, le *TextView* est passé à *setContentView()* pour l'afficher comme contenu de l'UI de l'*Activity*. Si l'*Activity* n'appelle pas cette méthode, il n'y aura pas d'UI et le système affichera un écran vide.

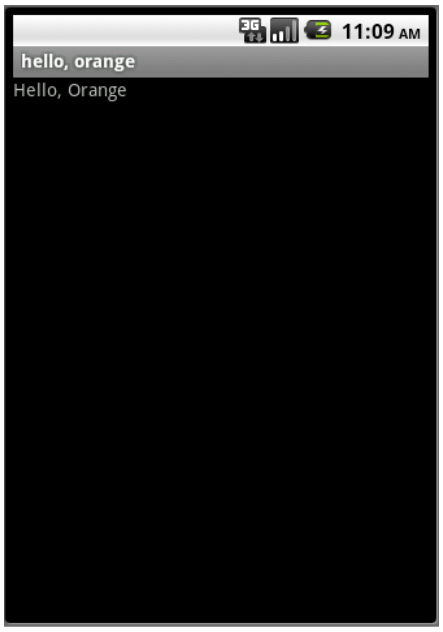
L'interface utilisateur est maintenant terminée, il faut l'exécuter pour l'afficher.

Exécution de l'application

Le plugin Eclipse facilite l'exécution des applications :

- Choisir **Run > Run**
- Sélectionner **Android Application**

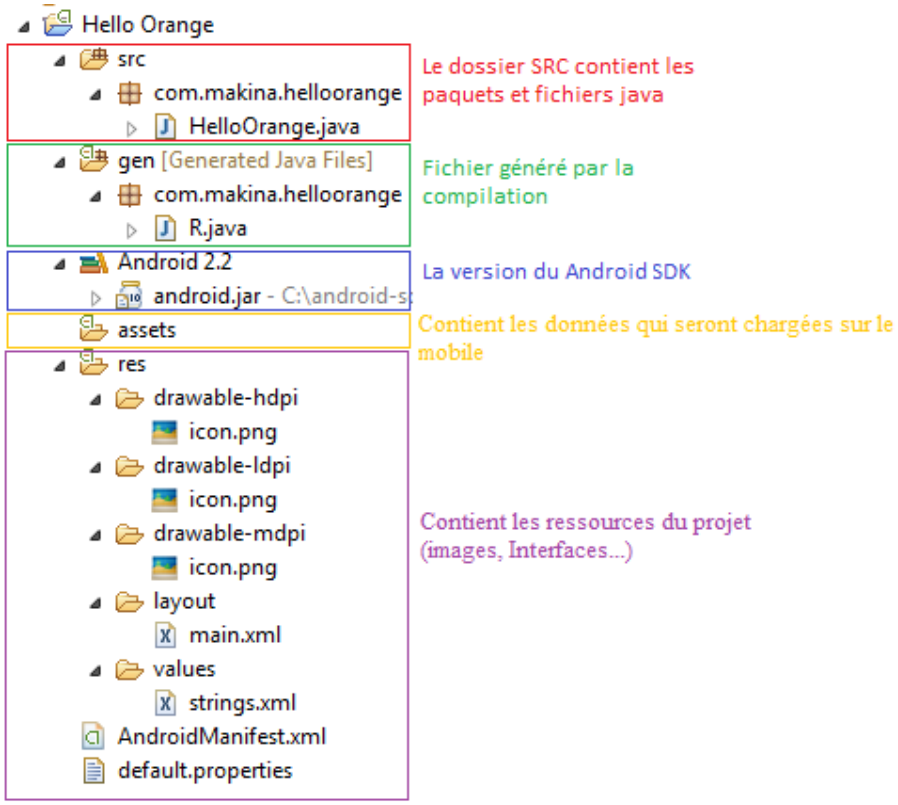
Le plugin Eclipse crée automatiquement une configuration d'exécution pour l'application et lance l'émulateur Android, plus ou moins rapidement suivant la configuration de la machine. Eclipse installe alors la nouvelle application et exécute l'Activity par défaut :



Le « hello, orange » apparaissant dans la barre grise est le titre de l'application.

Voici votre première application fonctionnelle !

Structure d'un projet sous Android



Répertoire src :

- C'est dans ce répertoire que l'ensemble des fichiers de code (.java) se trouvent.
- **Répertoire gen** : contient les fichiers générés par la compilation et qui ne sont pas modifiables par l'utilisateur.
- **Fichier R.java :** contient les références aux ressources du projet. Ainsi une image (drawable) est référencée par R.drawable.nom_image.
- **Répertoire Android *:*** SDK de la version d'Android choisie lors de la création du projet.
- **android.jar**. contient l'ensemble du framework Android.**Répertoire assets** : contient les données chargées sur le terminal lors de la compilation comme par exemple des fichiers texte de traduction, des sons, des images...
- **Répertoire *res*** : regroupe les éléments (ressources) du projet qui ne sont pas du code. Ce répertoire est parcouru par le framework pour créer R.java. Suivant leur nature, les ressources sont placées dans différents sous-répertoires :
  - **drawable** : pour les images bitmaps (png, parfois jpg ou gif)
  - **layout** : fichiers XML décrivant l'interface utilisateur
  - **values** : constantes comme des chaînes de caractères (strings.xml), des tableaux (arrays.xml), des couleurs (colors.xml), des booléens, des dimensions…
- **AndroidManifest.xml** : fournit au système Android des informations essentielles sur l'application. Ce fichier définit par exemple, le nom du package Java qui servira d'identifiant unique pour l'application,, la version minimale de l'OS, les bibliothèques externes, les composants de l'application (activités, services...)...

Distribution

Après avoir créé la prochaine *killer application* et l'avoir testée, il existe deux façons principales de la diffuser :

– créer un package apk que chacun pourra copier sur la carte mémoire SD de son terminal. Cette

méthode est réservée aux technophiles et est souvent utilisée pour tester une application en cours de développement

– publier sur l'Android Market.

L'Android Market est le point de rencontre entre développeurs et consommateurs qui utilisent la plate-forme Android. Vous pouvez y chercher des applications ou publier la vôtre.

Pour ce faire, n'oubliez pas de définir la version versionName et versionCode, l'icone icon et le nom label dans le fichier AndroidManifest.xml

Pour publier une application, il faut s'inscrire sur market.android.com/publish et payer 25$ de frais d'inscription. L'application peut être gratuite ou payante, générant ainsi des revenus pour le développeur.

communauté

La communauté est très active et vous aidera à amorcer votre apprentissage d'Android : de nombreuses ressources et de l'aide sont disponibles en ligne, notamment sur les sites suivants :

- Le site officiel de Google, qui est une véritable mine d'or (EN) : http://developer.android.com
- Le site d'entraide entre développeurs Stackoverflow (EN) : http://stackoverflow.com/questions/tagged/androidLa communauté francophone Android est également très active. Des sites comme http://www.androideur.com, http://android.cyrilmottier.com/ ou http://dev.frandroid.com présentent des tutoriels, des comparatifs... http://www.tunandroid.com/ en particulier présente des informations centrées sur le marché tunisien.

à retenir

	Très bon
Simplicité de développement	+ + + +
Cohérence de l'UI	+ +
Simplicité de distribution	+ + + + +
Utilisateurs potentiels	+ + + + +

Les plus

- Possibilité de distribution d'application hors Marketplace
- SDK sous forme de plugin de l'excellent Eclipse, multiplate-forme
- Simplicité du langage Java
- Personnalisation et intégration poussée des applications au sein de l'OS

Les moins

- Tests impératifs de l'application sur les nombreux appareils et écrans hétérogènes
- Pas d'outil simple de création d'UI

Jargon Buster :

API (Application Programming Interface) : interface logicielle permettant d'interagir avec une application ou les ressources d'un équipement, par exemple un téléphone mobile, son GPS, son accéléromètre, son appareil photo...

CSS : (**C**ascading **S**tyle **S**heets) : langage permettant de définir l'aspect graphique d'une page HTML (HyperText Markup Language) ou XML (Extensible Markup Language).

Eclipse : Environnement de Développement Intégré (IDE), libre et extensible, permettant de créer des projets de développement mettant en œuvre la plupart des langages de programmation. Écrit en Java, Eclipse est utilisable sur les OS où ce langage est disponible (Linux, Mac...)

Framework : espace de travail modulaire. C'est un ensemble de bibliothèques et de conventions permettant le développement rapide d'applications.

HTML (HyperText Markup Language) : format de données conçu pour représenter les pages web.

IDE (Integrated Development Environment) : programme regroupant un ensemble d'outils rendant plus aisé le développement d'applications (cf SDK).

Killer application : désigne une application informatique si attrayante qu'elle justifie à elle seule l'achat ou l'adoption d'un type particulier d'équipement : ordinateur, console de jeu, téléphone mobile...

OS (Operating System) : le système d'exploitation est l'ensemble de programmes centraux d'un appareil informatique servant d'interface entre le matériel et les logiciels applicatifs.

SDK (Software Development Kit) : le kit de développement ou trousse de développement logiciel est un ensemble d'outils permettant aux développeurs de créer des applications pour un matériel précis (iPhone, Android...)

Smartphone : téléphone mobile disposant des fonctions d'un assistant numérique personnel. Il peut ainsi fournir des fonctionnalités d'agenda, de calendrier, de navigation Web, de courrier électronique, de messagerie instantanée, de GPS... Les fonctionnalités des smartphones peuvent être enrichies par l'installation d'applications supplémentaires.

WAP (*Wireless Application Protocol*) : protocole de communication qui permettaient d'accéder à Internet à partir d'un téléphone mobile aux capacités limitées. Le WAP a perdu tout son intérêt à l'apparition des réseaux 3G et de terminaux plus puissants disposant d'un véritable navigateur web et permettant d'installer des applications supplémentaires.
Netbook : ordinateur portable léger et autonome, aux performances modestes et proposé à un prix réduit.

... Glossaire réalisé avec l'aide de Wikipédia : fr.wikipedia.org

Remerciements :

Ce guide a été réalisé avec l'aide des rédacteurs et contributeurs de la communauté de développeurs iPhone et Android de www.androideur.com, de www.tutomobile.fr, et de android.cyrilmottier.com.