UNIVERSITÉ DE NANTES

Université de Nantes

Logiciel Extensible

Onyx

Plateforme extensible

Étudiants :
Maxime Pauvert
Nicolas Bourdin
Yehudi Hollevoet
Tristan Jarry
Nicolas Brondin

Encadrant :
Gilles Ardourel

Mars 2015

Table des matières

Chapitre 1

Introduction

Dans le cadre du module de Conception de Logiciel Extensible, nous avons à réaliser un projet par groupes de cinq étudiants.

1.1 Objectifs

Le but de ce projet est de créer une plateforme pouvant accueillir des plugins, les charger, les lister, et les lancer, etc...

En plus de cette plateforme, nous devons également créer une application à partir de différents plugins, qui devront être lancés à partir de la plateforme.

1.2 Contraintes

— Langage de programmation : Java
— La plateforme doit être la plus simple possible
— L'application doit être compatible avec les machines du CIE
— Les plugins doivent pouvoir être simplement écrits par d'autres personnes

1.3 Application

Pour l'application fournie avec la plateforme, nous avons choisi de créer un système d'exploitation mobile très simplifié. Le but étant de pouvoir créer des applications et de les charger et les faire tourner sur le système avec des intéractions très simples.

Chapitre 2

Plateforme

2.1 Présentation

Onyx est une plateforme à plugins généraliste, ce qui veut dire que la plateforme seule n'a aucun comportement mais qu'en ajoutant des plugins respectants les interfaces fournies on peut créer n'importe quelle application et comportement.

On peut utiliser la plateforme de deux manières :

2.1.1 Lancement par défaut

Lorsqu'on lance la plateforme sans aucun paramètres, elle va automatiquement charger les plugins référencés dans le fichier "default-plugins.xml"

2.1.2 Lancement par commandes

Pour lancer des actions spécifiques de la plateforme, vous pouvez passer des commandes à l'execution. Si vous passez par maven, il faut passer les commandes comme ceci : -Dexec.args="-command"

"-help"

La commande help liste toutes les commandes disponibles sur la plateforme, idéal pour commencer à appréhender le fonctionnement de cette dernière.

"-pluginList"

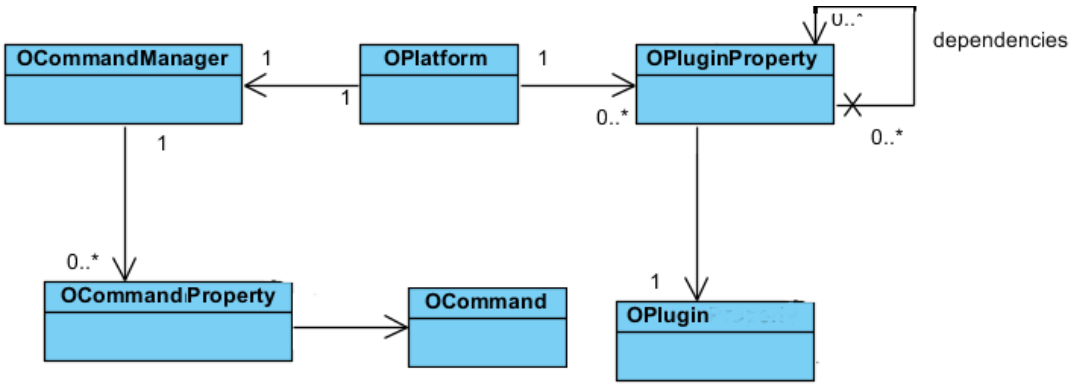
La commande pluginList liste tous les plugins disponibles sur la plateforme.

"-generatePlugin"

La commande generatePlugin permet de générer automatiquement un plugin, lorsque vous la tapez, elle vous affichera alors un menu ou vous rentrerez les infortations relatives à votre plugin.

2.2 Structure

2.2.1 Diagramme de classe



2.3 Installation

Tout d'abord, dans un terminal, tapez :

```
git clone https://github.com/masters-info-nantes/onyx.git
```

Lorsque tous les fichiers sont téléchargés,vérifiez que le proxy de maven est correct en faisant :

```
nano /.m2/settings.xml
```

Le fichier devrait ressembler à ça :

```
<settings>
    <proxies>
        <proxy>
            <id>example-proxy</id>
            <active>true</active>
            <protocol>http</protocol>
            <host>proxy.ensinfo.sciences.univ-nantes.prive</host>
            <port>3128</port>
        </proxy>
    </proxies>
</settings>
```

Placez-vous dans le dossier où vous avez cloné le projet. Tapez la commande :

```
mvn clean install
```

Pour copier tous les plugins dans la plateforme, tapez :

```
./mv_jars
```

Déplacez-vous ensuite dans le dossier onyx-platform :

```
cd onyx-platform/
```

Pour lancer l'execution par défaut, tapez :

```
mvn exec:java-X
```

2.4 Plugins

2.4.1 Création d'un nouveau plugin

Un plugin pour être intégré à la plateforme doit, comme vous vous en doutez, implémenter l'interface OPlugin comme ci-dessous :

myMainClass.java

```
package com.onyx.myPlugin;

import com.onyx.platform.OPlugin;
import com.onyx.platform.OPluginProperty;

public class myMainClass extends OPlugin{

    public myMainClass() {
        //Add some code here
    }

    @Override
    public void onCreate() {
        //Add some code here
    }

    @Override
    public void onStop() {
        //Add some code here
    }

}
```

Pour que votre plugin soit correctement reconnu par la plateforme, vous devez lui fournir un fichier OManifest sous cette forme : Attention ! Si vous définissez des dépendances pour le plugin, vous devez les inclure dans maven

OManifest

```
<?xml version="1.0"?>
<manifest>
    <id>com.onyx.myPlugin</id>
    <version>0.1</version>
    <name>myPlugin</name>
    <descriptionplugin for onyx</description>
    <mainClass>com.onyx.myPlugin.myMainClass</mainClass>
    /* les dépendences ne sont pas obligatoires */
    <dependencies>
        <dependency>
            <id>com.onyx.core</id>
            <version>${project.version}</version>
        </dependency>
    </dependencies>
```

```
</manifest>
```

Pour ceux qui voudraient se faciliter la vie et utiliser maven pour compiler leurs plugins, voilà un fichier pom.xml basique :

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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/x
    <parent>
        <artifactId>onyx</artifactId>
        <groupId>com.onyx</groupId>
        <version>1.0.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>myPlugin</artifactId>

    <dependencies>
        <dependency>
            <groupId>${project.groupId}</groupId>
            <artifactId>onyx-platform</artifactId>
            <version>${project.version}</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>${project.groupId}</groupId>
            <artifactId>onyx-core</artifactId>
            <version>${project.version}</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
</project>
```

2.4.2 Déclaration d'une commande

Chaque plugin peut apporter son lot de commandes avec lui, pour les définir, il suffit d'ajouter le code suivant dans le manifest :

```
<manifest>
    ...
    <command>
        <name>My Command</name>
        <id>mycommand</id>
        <usage>this command do some stuff</usage>
        <commandClass>com.onyx.myplugin.command.mycommand</commandClass>
    </command>
</manifest>
```

2.4.3 Déclaration d'un nouveau type de service

Tout comme les commandes, chaque plugin peut apporter son lot de services avec lui, pour les définir, il suffit d'ajouter le code suivant dans le manifest :

```
<manifest>
    ...
    <services>
        <service>com.onyx.core.OAppProperty</service>
    </services>
</manifest>
```

et la classe service devra contenir certaines annotations pour être interprété

2.4.4 Déclaration d'un service

Chaque plugin peut apporter son lot de service avec lui, l'ajout d'une commande est en réalisté l'ajout d'un service, son serviceType est quand à lui le seul présent de base dans la platforme. Ainsi le serviceType application est présent dans le core. Pour les définir, il suffit d'ajouter la description de la classe serviceType au manifest, voici un exemple pour le serviceType créé au dessus :

```
<manifest>
    ...
    <application>
        <id>mycommand</id>
        <mainActivity>com.onyx.myplugin.mainActivity</mainActivity>
    </command>
</manifest>
```

Chapitre 3

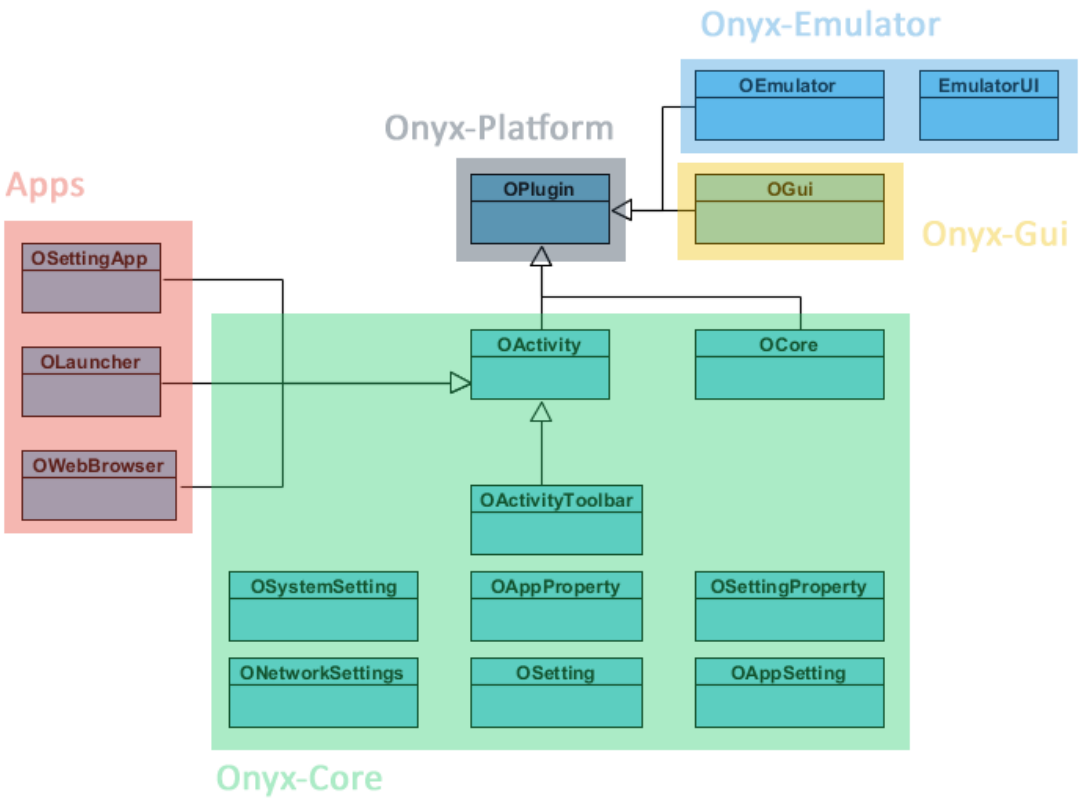
Application

3.1 Présentation

L'application se présente comme un système d'exploitation mobile simplifié fonctionnant grâce à quelques modules de base comme le core, l'interface graphique, un émulateur, tous étant remplaceables très facilement car ce sont en réalité des plugins branchés sur la plateforme Onyx avec plus ou moins de dépendances les uns entre les autres.

3.2 Structure

3.2.1 Diagramme de classes



3.3 Créer une application

Voici les fichiers et leurs contenu basique pour créer des plugins.

mainActivity.java

```
package com.onyx.myapppackage;
import com.onyx.core.OActivity;

public class OWebBrowser extends OActivity {
    @Override
    public void onCreate() {
        super.onCreate();
        //Some javafx code
    }
}
```

3.3.1 OManifest

```
<?xml version="1.0"?>
<manifest>
    <id>com.onyx.myapppackage</id>
    <version>${project.version}</version>
    <name>My App Package</name>
    <description></description>

    <application>
        <mainActivity>com.onyx.myapp.mainActivity</mainActivity>
        <name>My App</name>
        <id>com.onyx.myapppackage.myapp</id>
    </application>
</manifest>
```

3.3.2 pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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/x
    <parent>
        <artifactId>onyx</artifactId>
        <groupId>com.onyx</groupId>
        <version>1.0.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>onyx-app-my-app</artifactId>

    <dependencies>
        <dependency>
            <groupId>${project.groupId}</groupId>
```

```
            <artifactId>onyx-core</artifactId>
            <version>${project.version}</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
</project>
```

Chapitre 4

Conclusion