Java et systèmes embarqués

Jean-Francois Lalande - September 2012

Le but de ce cours est de découvrir les technologies Java permettant de développer des applications embarquées, notamment sur des téléphones portables. Ce cours permet de découvrir deux technologies supportées par Oracle, Java ME et Java FX. Une troisième partie aborde le développement d'applications clients serveurs en environnement embarqué.

1 Plan du cours

Plan du cours

Licence: la plupart des exemples de codes sont des extraits de code sources fournis par Oracle, sous licence L1 (cf fin du document).

2 Java ME

2.1 Les grandes familles de technologies Java

[EO] Java est à la fois un langage de programmation et une plate-forme de développement fournissant un environnement d'exécution rendant les applications indépendantes de la plate-forme. Les plate-formes étant multiples, il existe plusieurs editions des technologies Java:

- Java Platform, Standard Edition (Java SE): un environnement pour les applications de bureaux.
- Java Platform, Enterprise Edition (Java EE): un sur-ensemble de Java SE, orienté transaction et centré sur les bases de données (besoin orienté entreprise).
- Java Platform, Micro Edition (Java ME): environnement d'exécution et API pour les systèmes embarqués (téléphone, smartphone, assistants, TV).

Java ME n'est donc pas un nouveau langage de programmation. Il reste compatible avec Java SE autant que faire ce peut. Il peut définir de nouvelles interfaces ou APIs mais il tronque surtout une grosse partie de Java SE permettant d'alléger la machine virtuelle et les applications qui s'exécuteront sur un système embarqué très contraint.

Java ME se subdivise ensuite en 3 familles pour les systèmes embarqués:

- Java Micro Edition: téléphones, pages, pdas, ...
- Java Card: carte à puce
- Java SE Embedded: une version optimisée de Java ME

2.2 Les normes de Java ME

Java ME recouvre un ensemble de normes ou spécifications qui évoluent au cours du temps [MEDOC]. Sun (Oracle) a défini une numérotation JSR XXX qui correspond à chaque type de technologie qui constitue Java ME. Par exemple, la brique "RMI" est notée JSR 66, et les éléments du package java.security dédié à l'embarqué est numéroté JSR 219.

La page de référence qui décrit les différentes API [MEDOC] est située à:

http://www.oracle.com/technetwork/java/javame/documentation/apis-jsp-137855.html

On distingue dans Java ME trois grandes familles de technologies:

- CLDC: Connected Limited Device Configuration

- CDC: Connected Device Configuration
- Optional packages: bluetooth, sécurité, web service, graphiques, etc...

Les technologies de type CLDC définissent des spécifications permettant de développer des applications pour des périphériques nomades et à ressources limitées. Les spécifications définies dans CDC s'adressent à des applications distribuées, connectées au réseau et embarquées.

2.3 Profils et configurations

Java ME est découpé en 4 grandes parties:

- Les configurations (VM + Core Java APIs)
- Les profils (environnement de l'application)
- La partie applicative
- Les packages/APIs optionnels

Une configuration définit les possibilités de la machine virtuelle ainsi que l'API disponible pour le développeur. Une configuration est relativement indépendante du système embarqué cible. Elle définit une base commune à une famille de systèmes embarqués.

Un profil définit une API pour l'environnement de l'application c'est à dire le principe de fonctionnement et d'interaction entre le système embarqué et l'application.

Java Technologies for Handsets, Smart Cards, and Embedded Devices [EO]

2.4 Configuration CLDC

A l'heure actuelle, les configurations suivantes sont publiées par Oracle dans la catégorie CLDC:

- Connected Limited Device Configuration 1.0 JSR 30
- Connected Limited Device Configuration 1.1 JSR 139

La configuration CLDC 1.1 définit l'ensemble des classes fournies par un machine virtuelle compatible. Le nombre de package étant très limité, il est même possible de les lister exhaustivement:

```
java.io: I/O
java.lang: classes fondamentales (types, exceptions, erreurs)
java.lang.ref: weak references
java.util: collections et dates
javax.microedition.io: définition des interfaces de connection
```

Ces packages regroupent un sous ensemble restreint de classes de la spécification J2SE 1.4.

java.io

Interfaces

```
DataInput
DataOutput
```

Classes

```
ByteArrayInputStream ByteArrayOutputStream
DataInputStream DataOutputStream
InputStream InputStreamReader
OutputStream OutputStreamWriter
PrintStream
Reader Writer
```

Exceptions

```
EOFException
InterruptedIOException
IOException
UnsupportedEncodingException
UTFDataFormatException
```

java.lang

Interfaces

```
Runnable
```

Classes

```
Boolean Byte Character Class Double Float
Integer Long Math Object Runtime Short
String StringBuffer System Thread Throwable
```

Exceptions

```
Exception ArithmeticException ArrayIndexOutOfBoundsException
ArrayStoreException ClassCastException ClassNotFoundException
IllegalAccessException IllegalArgumentException
IllegalMonitorStateException IllegalThreadStateException
IndexOutOfBoundsException InstantiationException
...
```

Errors

```
Error NoClassDefFoundError OutOfMemoryError VirtualMachineError
```

java.lang.ref

Classes

```
Reference
WeakReference
```

Apparté sur les weak références

Il s'agit de permettre au programeur de créer un pointeur sur un objet qui n'empêchera pas le *garbage collector* de collecter l'objet pointé par une *weak reference*. On construit un objet de la classe **WeakReference** en passant au constructeur l'objet pointé. Il y a donc une sorte d'indirection puisque le programmeur possède un pointeur sur la *weak reference* qui elle même pointe vers l'objet. Cependant, l'implémentation des *weak references* signale à la machine virtuelle que l'objet est pointé par une référence dite "faible" et qui peut donc laisser le *garbage collector* agir.

```
public WeakReference(Object ref)
public Object get(); // Returns this reference object's referent
public void clear(); // Clears this reference object.
```

java.util

Interfaces

```
Enumeration
```

Classes

```
Calendar
Date
Hashtable
Random
Stack
TimeZone
Vector
```

Exceptions

```
EmptyStackException
NoSuchElementException
```

javax.microedition.io

Interfaces

```
Connection
ContentConnection
Datagram
DatagramConnection
InputConnection
```

Ceci est un exemple, cliquez sur le lien de téléchargement pour obtenir le cours complet.