Java ME : une présentation

Jean-Marc Farinone

But de l’exposé

- Comprendre, définir, situer les termes :
  - Java ME, J2ME, CDC, CLDC, Configuration, Profiles, MIDP (1.0, 2.0), DoJa, MIDlet, jad, etc.
  - Donner des références
  - Donner des exemples de programmes

Les concepts fondamentaux de Java ME

Java ME = ?

- Java ME = Java Micro Edition
- Site de référence = `http://java.sun.com/javame/index.jsp`
- S'appelait anciennement J2ME : terme encore très souvent employé.
- Famille de spécifications pour développer des logiciels pour des objets électroniques (device = périphérique) comme

  - les téléphones portables,
  - les assistants personnels (PDA)

  - « téléphones intelligents » (smart phones)

J2ME = Configuration et Profiles

- Le monde des périphériques électroniques est vaste, divers et varié.
- => Pas de possibilités d'avoir un seul environnement uniforme pour tous (!= J2SE)
- => Architecture en couche :
  - Bibliothèques de base : les configurations
  - Les ajouts à ces bibliothèques : les profiles

Configuration

- = Spécifications pour un ensemble de périphériques ayant des caractéristiques similaires comme :
  - Le type et la quantité mémoire disponible
  - Le type de processeur (vitesse, etc.)
  - Le type de réseau disponible pour ce périphérique
- Configuration = plate-forme minimale pour cet ensemble. Pas d'extension ni de retrait possible
- => portabilité

Les deux configurations fondamentales

- CLDC (Connected Limited Device Configuration), CDC (Connected Device Configuration)
- CLDC ~ wireless Java.
  - Pour téléphone cellulaire, PDA ayant 192 Ko de mémoire minimum (CLDC 1.1) pour la JVM
  - Téléchargement de programmes Java
  - 2 versions 1.0 (JSR-30 Mai 2000), 1.1 (JSR-139 Mars 2003)

Les deux configurations fondamentales (suite)

- CDC = entre CLDC et J2SE
  - Périphériques ayant 2Mo ou plus de mémoire : smart phone, web téléphone, boitier TV (set-top boxes).

Configuration =

- Une JVM + environnement Java (paquetages, outils, etc.)
- JVM J2ME < JVM J2SE
  - Exemple CLDC 1.0 JVM n'a pas de type `float`, `double`.
  - Mais CLDC 1.1 et CDC VM les a !
- Les spécifs J2ME indiquent souvent ce qu'elles n'ont pas % J2SE.

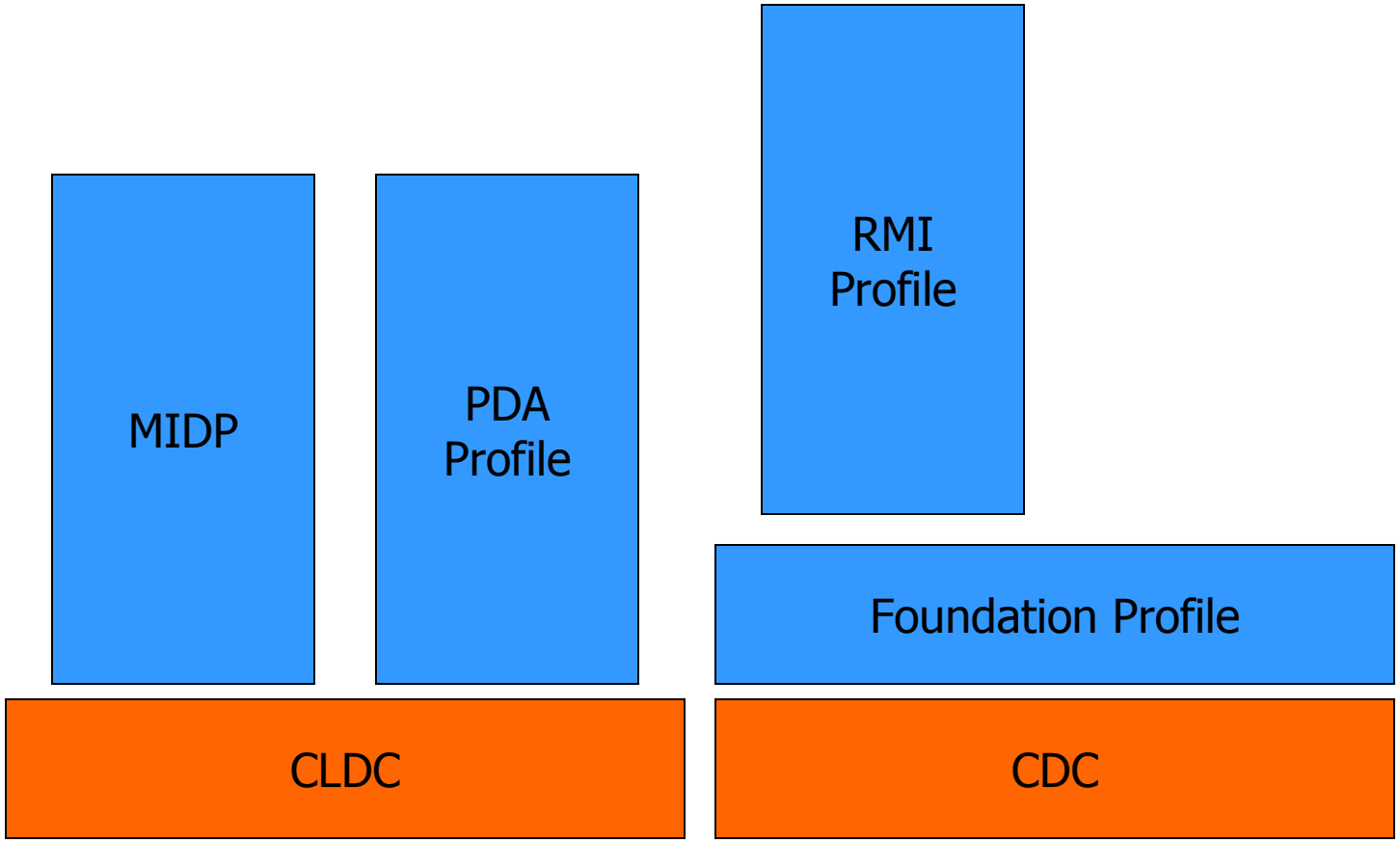
Configuration CLDC =

- KVM avec un ramasse-miettes
- Premières implémentations : SUN (Win32, Solaris, Linux) et J9 VM d’IBM

Profile

- = un complément à une configuration.
- Apporte des classes supplémentaires pour un domaine ou un marché spécifique
- Les profiles au dessus de CLDC :
  - MIDP (Mobile Information Device Profile)
  - PDA Profile
- Les profiles au dessus de CDC :
  - Foundation Profile
  - RMI Profile

Configuration et Profile : conclusion



MIDP =

- Mobile Information Device Profile
- Amène :
  - la partie réseau (+ HTTP 1.1)
  - des composants d'IHM
  - le stockage local
- Disponible sur PalmOS

PDA Profile =

- Ecran plus puissant
- Donc IHM plus riche
- Accessibilité aux bibliothèques natives

Foundation Profile =

- Permet d'avoir presque tout J2SE 1.3
- Le profile de départ pour les autres profils

RMI Profile =

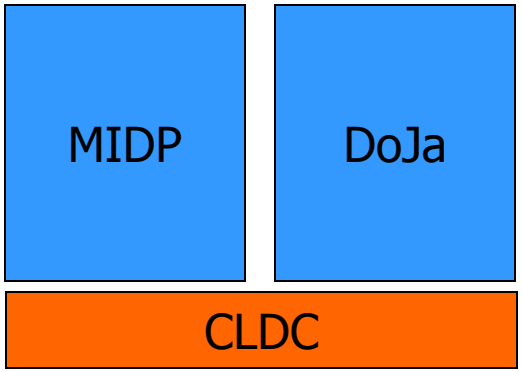
- Amène la partie cliente de RMI
- Voir à `http://jcp.org/en/jsr/detail?id=66`
- Utilise TCP/IP
- En vue de faire du Jini sur périphériques

DoJa =

- DoCoMo Java de NTT-Docomo (Japon)
- Pour «tel portable puissant» : écran couleur, réseau GPRS, son polyphonique, HTTPS, etc., et autres périphériques (consoles de jeux, etc.)
- En France, I-mode (= internet + tel) (Bouygues)
- Au dessus de CLDC (donc // à MIDP)
- Site référence (Bouygues) : `http://communaute.imode.fr/p21.php`

DoJa (suite)

- DoJa fournit plus de fonctionnalités que MIDP :
  - plus de composants graphiques,
  - HTTPS et SSL (business oblige)
- Environnement de développement amené par Bouygues et DoCoMo
- Finalement



J2ME : les restrictions % J2SE

- Règles fondamentales :
  - Une interface de programmation qui existe dans J2SE et qui est repris dans J2ME se trouve dans le même paquetage, la même classe avec la même sémantique que dans J2SE
  - Il peut y avoir des champs et méthodes en moins dans une classe
- Les notions propres à J2ME en ajout % J2SE se trouvent dans des paquetages autres que ceux de J2SE.

CLDC : les restrictions % J2SE (suite)

- 3 paquetages repris (pas en totalité) :
  - `java.io`, `java.lang`, `java.util`
- Des paquetage additionnels, sous paquetages de `javax.microedition`

Configuration CDC =

- JVM pour CDC = CVM = JVM de J2SE 1.3 sans le JIT compiler
- Implémentation pour Linux/x86 et VxWorks (temps réel).
- Rien pour Windows ?
- Voir à

  `http://java.sun.com/products/cdc/index.jsp`
- S’inscrire au Sun Download Center (SDLC)
- Makefile pour outils gnu (gcc, …)

MIDP

Introduction

- Pas d'APIs d'interaction utilisateur, de stockage, de réseau, dans CLDC
- d'où MIDP
- applications MIDP = MIDlets
- réseau par HTTP 1.1 au moins (pas forcément TCP/IP)

IHM MIDP

- IHM sur un "petit" écran :
  - au moins 96 pixels de large sur 54 pixels de haut,
  - 2 couleurs,
  - rappel !! PDA = 160x160, 65536 couleurs
- "petit" clavier 

ou 

  au moins les chiffres de 0 à 9, les flêches, un bouton de sélection (ou équivalents).

`jar`, `jad` et cie

- Les MIDlets et leur ressources sont mises dans un `.jar`
- … qui peut être très gros
- Le contenu du `.jar` est décrit par son fichier `META-INF\MANIFEST.MF` (comme d'hab)
- Pour éviter d'avoir à charger tout le `.jar` pour avoir des infos sur l'archive (et éventuellement alors l'ignorer !!) une copie du manifeste est créée et peut être chargée : le `.jad`

Exemple de `jad`

- Rappel : le format d'un `jad` est celui d'un fichier manifeste.

`FPDemo.jad`

```
MIDlet-1: Calculator, calculator.png, calculator.CalculatorMIDlet
MIDlet-Description: Floating Point demonstration MIDlet
MIDlet-Jar-Size: 2451
MIDlet-Jar-URL: http://www.monSite.fr/FPDemo.jar
MIDlet-Name: FPDemo
MIDlet-Vendor: Sun Microsystems, Inc.
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.1
MicroEdition-Profile: MIDP-2.0
```

- Champs importants :
  - `MIDlet-Jar-Size: 2451`
  - `MIDlet-Jar-URL: http://www.monSite.fr/FPDemo.jar`

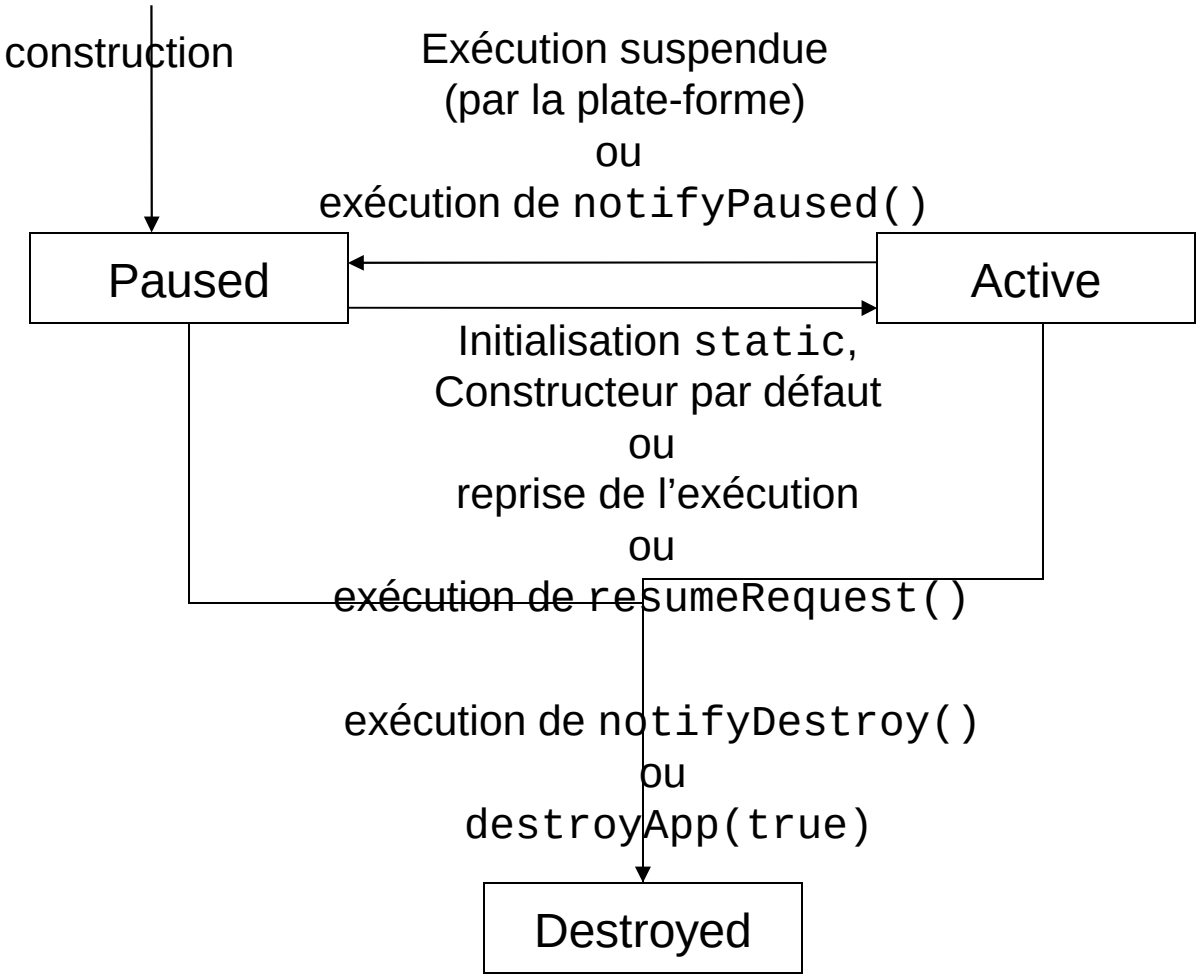
MIDlet

- Dérive de la classe abstraite `javax.microedition.midlet.MIDlet`
- Doit avoir un constructeur par défaut (éventuellement donné par le compilateur)
- La MIDlet minimale :

```
// pour la classe abstraite MIDlet
import javax.microedition.midlet.*;

public class TrameMIDletJMF extends MIDlet {
    // les 3 méthodes abstraites de MIDlet
    public void destroyApp(boolean condition)
         throws MIDletStateChangeException {}
    public void pauseApp() { }
    public void startApp() throws MIDletStateChangeException {}
    public TrameMIDletJMF(){ }
}
```

MIDlet : son cycle de vie



Développer une MIDlet

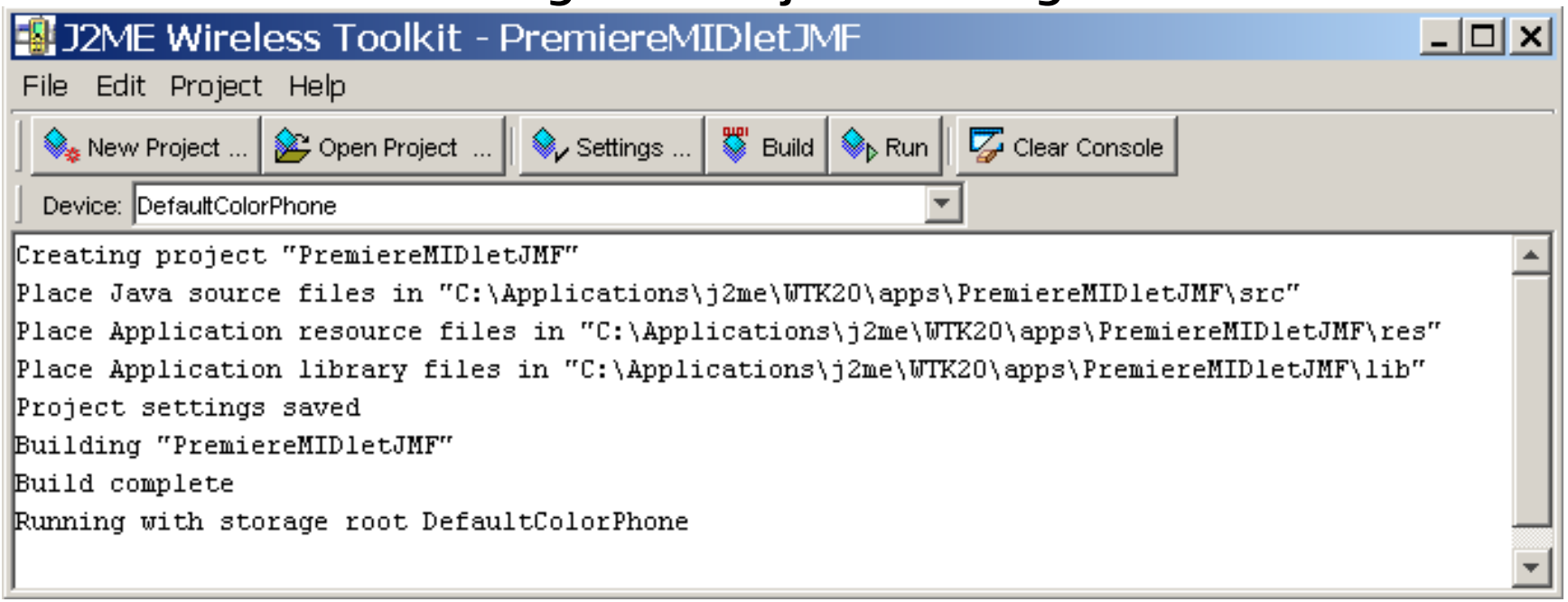
- Installer J2SE.
- Charger gratuitement l'environnement "Wireless toolkit" à partir de `http://java.sun.com/products/j2mewtoolkit/index.html`
- Eventuellement être inscrit au Download Center de Sun.
- Des commandes en ligne existent

Développer une MIDlet (suite)

- Lancer la Ktoolbar
- Créer un projet (New Project). Donner un nom de projet, le nom de la classe MIDlet. Cliquer "Create Project". =>
  - 1°) Les infos du `.jad` sont affichées.
  - 2°) un répertoire du nom du projet a été créé sous l'environnement wireless toolkit.
- Placer sources, ressources, etc. dans ce répertoire.
- Début de la demo

Développer une MIDlet (suite)

- Cliquez Build. L'environnement a :
  - Créer les répertoires `classes`, `tmpclasses`.
  - Compiler les sources Java, résultat dans `tmpclasses`
  - Prévérifier ces `.class` et mis dans `classes`
  - Construit les `.jar` et ajuste le `.jad`



Développer une MIDlet (fin)

- Au fait, code de la MIDlet !

```
import javax.microedition.midlet.*;
// pour CommandListener
import javax.microedition.lcdui.*;
public class PremiereMIDletJMF extends MIDlet implements CommandListener {
    // les 3 méthodes abstraites de MIDlet
    public void destroyApp(boolean condition){}
    public void pauseApp(){}
    public void startApp(){
         Display.getDisplay(this).setCurrent(mMainForm);
    }
    // La methode de l'interface CommandListener
    public void commandAction(Command c, Displayable d) {}
    public PremiereMIDletJMF() {
         mMainForm = new Form("Ma Premiere MIDlet JMF");
         mMainForm.append(new StringItem(null, "Bonjour à tous"));
         mMainForm.addCommand(new Command("Exit", Command.EXIT, 0));
         mMainForm.setCommandListener(this);
    }
    private Form mMainForm;
}
```

Exécuter la MIDlet

- Cliquer "Run"
- Changer de périphérique par Device (QwertyDevice)
- Une demo : OK !
- Plus de code ...
- … au prochain exposé (programmation réseau avec MIDP)

IHM et architecture d'une MIDlet

- En général, on prépare tout dans la MIDlet :
  - les divers écrans qui risquent d'apparaître
  - les divers `Command` utiles à ces écrans
- Puis on fait en sorte que la MIDlet soit auditeur de ces `Command`
- Ainsi lorsque l'utilisateur appuie sur une `Command`, la main est repassée à la MIDlet qui redirige vers le prochain écran.

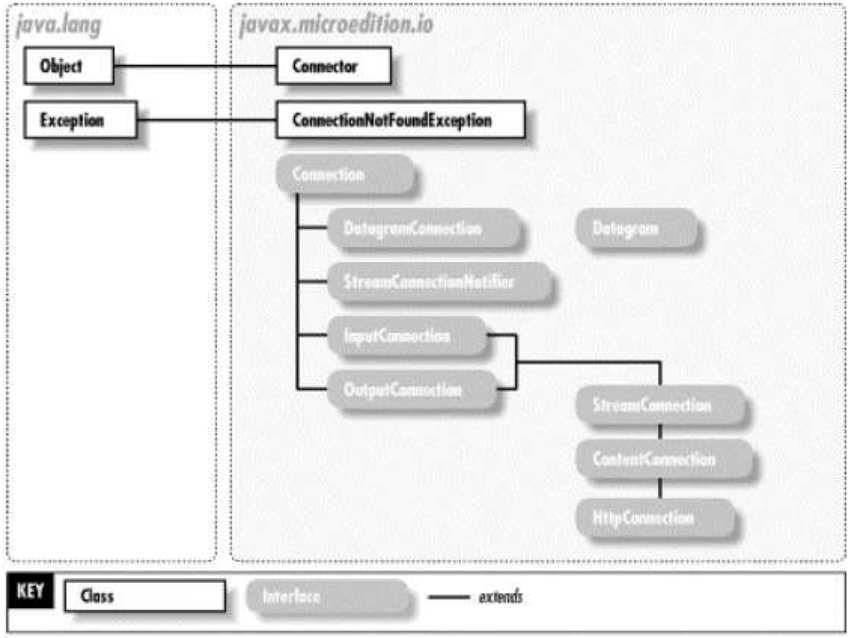
Programmation réseau pour Java ME (CLDC, MIDP)

Jean-Marc Farinone

Présentation

- J2SE qui contient des classes pour les protocoles TCP, UDP, IP (`java.net`) et aussi RMI, CORBA, JINI, etc.
- En J2ME, CLDC/MIDP 1.0, on ne peut avoir tout cela.
- La programmation réseau de CLDC est gérée par le Generic Connection Framework (GCF)
- GCF utilise le package `javax.microedition.io`.

Classes, interfaces, exceptions de `javax.microedition.io`



- source J2ME in a nutshell (Kim Topley, ed O'Reilly)
- Donc finalement essentiellement des interfaces !

Les protocoles de communication pour mobiles

- Les mobiles (en fait les fournisseurs de réseau pour mobiles) ne fournissent pas toujours de connection socket ou TCP (les émulateurs si ;-))
- Mais cela peut exister
  - sur les PDA
  - en payant (très) cher
- Ce qui est supporté est l'accès HTTP.

`Connection` and `Connector`

- `Connection` est une interface
- `Connector` est une classe
- `Connector` fournit les connexions et `Connection` les referment.
- Concrêtement, la classe `Connector` possède des méthodes statiques qui rendent des connexions et `Connection` possède la méthode `close()`.

La classe `Connector`

- "This class is a factory for creating new `Connection` objects."
- Cette classe ne possède que des méthodes statiques qui permettent de récupérer des connexions (pour certains protocoles) ou des canaux de communications
- Méthodes proposées
  - `public static Connection open(String name) throws IOException`
    où `name` est une URL sur laquelle se connecter
  - Par exemple pour `name` :
    - socket://www.amazon.com:80
    - http://www.cnam.fr/MaBelleServlet
    - comm:0;baudrate=28800;parity=even

Obtenir une connexion avec `Connector`

- On utilise une des méthodes de cette classe :

```
public static Connection open(String name)
throws IOException
public static Connection open(String name, int
mode) throws IOException
public static Connection open(String name, int
mode, boolean timeouts) throws IOException
```

- name indique le "type" de connexion demandée. Sa syntaxe est une URL de la forme : `{scheme}:[{target}][{params}]` avec :
  - `{scheme}` un protocole
  - `{target}` des informations réseau
  - `{params}` est une suite de la forme `";variable=valeur"`

Méthodes de la classe `Connector`

- On a plus souvent :
  - `public static Connection open(String name, int mode) throws IOException`
    où `mode` est un constante `READ`, `READ_WRITE`,`WRITE` de cette classe `Connector`
- Voire
  - `public static Connection open(String name, int mode, boolean timeouts) throws IOException`
    où si `timeouts = true`, une exception est levée lorsque délai de connexion inactive est dépassé (si cela est supporté par le mobile !!)
- name (l'URL) est de la forme :
  - `protocole:adresse;paramêtres`

Plus précisément , arguments de `open()`

- `mode` indique si la connexion permettra de faire des lectures, des écritures ou les deux. Ses valeurs sont les constantes statiques de cette classe : `READ`, `WRITE`, `READ_WRITE` (valeur par défaut)
- `timeouts` un boolean qui, si sa valeur est `true`, permet de lever une exception lorsque le délai de garde (dépendant du protocole) est dépassé.

Autres méthodes de la classe `Connector`

- On peut avoir des méthodes qui retournent directement un flot comme :
  - `public static DataInputStream openDataInputStream(String name) throws IOException`
  - `public static InputStream openInputStream(String name) throws IOException`
  - `public static DataOutputStream openDataOutputStream(String name) throws IOException`
  - `public static OutputStream openOutputStream(String name) throws IOException`
- name est encore l'URL

Les connexions retournées

- En général, on essaie d'avoir une seule connexion qui permet de lire et écrire.
- Ceci est obtenu par une `StreamConnection` qui est une interface qui hérite de `InputConnection` et `OutputConnection`
- Les méthodes de ces interfaces implémentées par des classes vont retourner des `DataInputStream` et `DataOuputStream` et donc on peut ainsi utiliser les méthodes de `DataInputStream` et `DataOuputStream`.

Plan pour une communication socket

- 1°) Obtenir une connexion
- 2°) Récupérer les canneaux d'écriture et de lecture sur cette connexion
- 3°) Envoyer une requête, récupérer et traiter la réponse
- 4°) Fermer la connexion
- Remarque : ce plan fonctionne si c'est le mobile qui est initiateur de la connexion

1°) Obtenir une connexion

```
StreamConnection socket;
try {
    String server = ...
    String port = ...
    String name = "socket://" + server + ":" + port;
    socket = (StreamConnection)Connector.open(name,
Connector.READ_WRITE);
} catch (Exception ex) { ... }
```

2°) Récupérer les canneaux d'écriture et de lecture sur cette connexion

```
OutputStream os = null;
InputStream is = null;

os = socket.openOutputStream();
is = socket.openInputStream();
```

3°) Envoyer une requête, récupérer et traiter la réponse

```
String request = "GET / HTTP/1.0\n\n";
os.write(request.getBytes());

...

// On lit au plus 128 octets
final int MAX_LENGTH = 128;
byte[] buf = new byte[MAX_LENGTH];
int total = 0;
while (total < MAX_LENGTH) {
    int count = is.read(buf, total, MAX_LENGTH - total);
    if (count < 0) { break; }
    total += count;
}

String reply = new String(buf, 0, total);
```

4°) Fermer la connexion et demo

```
os.close();
is.close();
socket.close();
```

- source Java in a Nutshell, Kim Topley
- demo socket dans le WTK, projet Chapter 6, MIDlet socket. Le vérifier :
  - en lançant timcat localement
  - en se connectant dans la MIDlet socket sur le serveur `localhost` port `8080`,
  - en le vérifiant par `telnet localhost 8080` et lancée de la requête **`GET / HTTP/1.0\n\n`**

Le protocole HTTP

- Le seul protocole réellement supporté dans MIDP
- Car en général, les mobiles n'ont pas de communication directe à "l'internet (TCP)"
- rappel : HTTP est un protocole requête réponse
- Il suffit d'indiquer l'URL et chaque `getXXX()` récupère immédiatement la réponse.

Programmation pour le protocole HTTP

```
HttpConnection hc = null;
InputStream in = null;
String url = "http://localhost:8080/midp/hits";

try {
  hc = (HttpConnection)Connector.open(url);
  in = hc.openInputStream();

  int contentLength = (int)hc.getLength();
  byte[] raw = new byte[contentLength];
  int length = in.read(raw);

  in.close();
  hc.close();

  // traite la réponse
  String s = new String(raw, 0, length);
  ...
}
catch (IOException ioe) { ... }
```

MIDP et HTTP : une démo dans eclipse

- Voir à `http://developers.sun.com/mobility/midp/articles/tutorial2/` d'une communication mobile sur une servlet car Jonathan Knudsen
- Excellent tutorial les divers API de MIDP 2.0 à `http://developers.sun.com/mobility/learn/midp/midp20/`

Bibliographie

- `http://developers.sun.com/mobility/getstart/` : une présentation des diverses JSR de J2ME
- J2ME in a nutshell. Kim Topley ; éditions O'Reilly
- J2ME Wireless Toolkit 2.1 Download à http://java.sun.com/products/j2mewtoolkit/download-2
  J2ME, applications pour terminaux mobiles. Bruno Delb ; éditions Eyrolles
- http://java.sun.com/products/cldc/index.jsp : page initiale de CLDC

Bibliographie

- Java development on PDAs. Daryl Wilding-McBride ; éditions Addison-Wesley
- J2ME in a nutshell. Kim Topley ; éditions O'Reilly
- J2ME Wireless Toolkit 2.1 Download à

  `http://java.sun.com/products/j2mewtoolkit/download-2_1.html`
- J2ME, applications pour terminaux mobiles. Bruno Delb ; éditions Eyrolles
- `http://java.sun.com/products/cldc/index.jsp` : page initiale de CLDC

Bibliographie (suite)

- `http://communaute.imode.fr/p21.php` et `http://www.cellconcept.com/faq_doja.html` pour I-mode, DoCoMo

Fin