Université Joseph Fourier
PolyTech & IMA

Année Universitaire 2003-2004
RICM3 SR & M2GI SRR
Communications Mobiles

Concepteur : Didier DONSEZ <didier.donsez@imag.fr>
But : Manipuler J2ME

PRATIQUE DE J2ME

Le but de ce TD est de tester l'environnement J2ME.

Installation du J2ME Wireless Toolkit

Normalement, vous téléchargez de du site http://java.sun.com/products/j2me et l'installer.
Cependant vous n'avez qu'à décompresser le fichier suivant : http://www-adele.imag.fr/~donsez/cours/cm/wtk21.zip
Configurez JAVA_HOME et ANT_HOME
Configurez WTK_HOME (set WTK_HOME=C:\TEMP\WTK21)
Ajoutez %WTK_HOME%\bin à votre PATH

Les commandes suivantes vous permettent de :

- `defaultdevice` : configurer le mobile de test (téléphone, pager, …) . Palm requiert l'émulateur POSE et une ROM PalmOS !
- `emulator` : lancer l'émulateur d'un mobile (l'option –Xdevice :<*device*> précise le type de mobile ex : -Xjam)
- `prefs` : configurer quelques paramêtres (proxy, …)
- `ktoolbar` : lancer l'éditeur/lancer de projet MIDlet
- `preverify` : vérifie les contraintes J2ME sur les classes Java lors de la construction du projet

La documentation est accessible depuis %WTK_HOME%\index.html

Prise en main i

Lancez `ktoolbar`
Ouvrez un des projets (*games* par exemple)
Les projets (présents dans %WTK_HOME%\apps)sont brièvement décrit dans %WTK_HOME%\docs\AppDemos.html
Construisez le [*Build*]
Exécutez le [*Run*] et testez l'interface homme-machine (L'émulateur que vous avez choisi est lancé avec les MIDlets du projet)
Visualisez les 2 scripts build.sh et run.sh dans du répertoire %WTK_HOME%\apps*\bin\
Certains applications ont des scripts ANT !

Prise en main ii

Lancez l'émulateur avec un des .jad situé dans %WTK_HOME%\apps*\bin\

Développement

Ajoutez le répertoire Z:\J2ME_ENS\sujet\apps-to-add* dans votre répertoire %WTK_HOME%\apps
Lancez `ktoolbar`
Créez [*New Project*] un nouveau projet `hello` avec la classe `examples.hello.HelloMIDlet`
Configurez [*Settings*] la MIDlet et notamment son icône `/examples/hello/icon/hello.png`
Générez [*Build*] les classes
Exécutez dans le simulateur [*Run*]

Exécutez avec un autre mobile (*device*)
Procédez de même avec animation, addressbook, http, xml (attention, l'exemple xml nécessite les packages de kXML (http://www.kxml.org et sur ~/J2ME/kxml)
Utilisez un ofuscateur pour reduire la taille des MIDlets
Aidez vous de la documentation %WTK_HOME%\index.html

Débogage

Lancez ktoolbar
Ouvrez le projet addressbook
Positionnez l'option de deboggage [Project>Debug]
Reconstruisez le projet
Lancez l'application [Run]
Lancez le déboggeur jdb sur le port de deboggage

```
jdb -connect com.sun.jdi.SocketAttach:hostname=localhost,port=5000 \
       -sourcepath %WTK_HOME%\apps\addressbook\src
```

Aidez vous de la documentation de jdb

Développement suite

Regroupez les 4 exemples précédents sous un seul projet total qui contiendra les 4 MIDlets.
Ecrivez un script (`build.sh`, `build.bat ou build.xml`) qui construit `total.jar`
Configurez le fichier `total.jad` avec `ktoolbar`
Testez [*Run*]

Déploiement

Installez `total.jar` et `total.jad` dans le répertoire `midlets` à la racine d'un serveur HTTP (celui de GICOM par exemple)
Changez la propriété `MIDlet-Jar-URL` de `total.jad` avec l'URL absolu de JAR `http://monserverweb:8080/midlets/total.jar`
Assurez vous que `webapps/midlets/WEB-INF/web.xml` contiennent les correspondances suivantes pour les types MIME

```
jad        text/vnd.sun.j2me.app-descriptor
jar        application/java-archive
```

Lancez l'émulateur sans passer par `ktoolbar` avec la commande suivante :

```
emulator -Xdescriptor:http://monserverweb:8080/midlets/total.jad &
```

Installez les MIDlets sur l'émulateur en passant l'URL absolue de total.jad

```
emulator -Xjam:install=http://monserverweb:8080/midlets/total.jad &
```

Vous pouvez changer le device avec la commande suivante `DefaultDevice`

Emulation PalmOS

(sous Windows seulement et avec WTK 1.0.4)

Installation de POSE

Normalement, vous téléchargez l'émulateur Palm OS (POSE) sur le site développeur http://www.palmos.com
Cependant vous n'avez qu'à copier le répertoire Unix ~donsez/PALM dans votre répertoire ~/PALM.
Sous Windows, exécutez l'émulateur POSE/PalmOS-Emulator.exe
Appuyez sur New pour démarrer une nouvelle session d'émulation.
Configurez la ROM que vous sélectionnez dans Z:\PALM/ROM.
Désactivez toutes les options de deboggage dans le « menu bouton droit> Settings > Debug Options »

MIDlet sur émulateur POSE

Lancez ktoolbar.bat
Ouvrez un projet (demos)
Sélectionnez le device PalmOS_device puis lancez [Run]
Indiquez l'exécutable /PALM/POSE/PalmOS-Emulator.exe
Testez la MIDlet

Emulation DoJa

DoJa est un profile basé sur le J2ME/CDLC/MIDP pour les téléphones mobiles Docomo iMode.
Téléchargez et installez le kit de développement et d'émulation DoJa.
Recommencez les tests effectués précédemment avec le WTK.

Développement avec eCOM

L'objectif de cet exercice est d'ajouter à votre site de commerce électronique eCOM que vous avez développé d'Octobre à Décembre, la possibilité d'accepter des clients J2ME/CDLC/MIDP !
Quel peut être l'intérêt d'utiliser J2ME plutôt que WAP/WML pour ce type d'application ?

> *La MIDlet peut être autonome : le caddie est entièrement géré sur le terminal*

Basez vous sur le code de `examples.http.DownloadMIDlet` pour écrire une MIDlet `ecom.midlet.EcomMIDlet` (et la servlet correspondante) qui consulte les catalogues des magasins du projet Ecom.
Complétez `EcomMIDlet` pour qu'elle puisse constituer un caddie embarqué (géré au niveau du terminal).
Modifiez `EcomMIDlet` pour qu'elle envoie le contenu du caddie à la servlet `ProcessCartServlet` sous la forme d'un document XML (HTTP POST))
Ecrivez la servlet `ProcessCartServlet` qui reçoit toutes les entrées du caddie et procède au paiement du caddie.

Documentation

```
%WTK_HOME%\index.html
http://www-adele.imag.fr/~donsez/cours/j2me.pdf
http://developer.java.sun.com/developer/J2METechTips/index.html
http://wireless.java.sun.com/midp/articles
http://www.kvmworld.com
```

Jeu de la Vie (Automate cellulaire de Conway)

Ecrire une MIDlet ConwayMIDlet réalisant une animation basée sur le jeu de la vie.
Rappel des règles de Conway:

- Une cellule se crée s'il y a 3 cellules voisines (parmi 8)
- Une cellule reste vivante s'il y a 2 ou 3 voisines
- Sinon elle meurt

Remarque : plusieurs contenants (collection) peuvent être utilisés : HashMap, Array, Tree, LinkedList, … pour représenter la population avec des performances et des consommations mémoire variables ! Quel contenant choisir ?
Vous étudierez les performances au moyen du profiler du J2MESDK
Indications : lire Jonathan Allin, Wireless Java for Symbian Devices, http://www.symbian.org/books et http://servlet.java.sun.com/javaone/resources/content/sf2002/conf/sessions/pdfs/1022.pdf page 46

Mini-Projet « MiniPIM »

PIM est l'acronyme de Personal Information Manager. C'est un ensemble d'applications de base des terminaux nomades. Cet ensemble inclut généralement : un carnet d'adresse, un agenda, un bloc note, un gestionnaire de mots de passe, …)

Dans ce mini-projet, vous développerez l'application de carnet d'adresse.
Vous pourrez partir de l'exemple AddressBook étudié précédemment

Etape 1 : Les entrées du carnet

Cette étape consiste à visualiser/ajouter les entrées du carnet.
Chaque entrée a plusieurs champs qui peuvent être multivalués (exemple : plusieurs numéros de téléphone). Des champs de bases (firstname, lastname, phone, gsm, address, city, zipcode, country) sont prédéfinis cependant l'utilisateur peut en ajouter. Tous les champs sont de type texte .

Etape 2 : Recherche et Filtres (dépendance 1)

Cette étape consiste à rechercher les entrées du carnet sur plusieurs critères.
La recherche utilise des filtres simples (débute par la sous-chaîne, contient la sous-chaîne, tous les champs, un de ces champs).
Les filtres sont mémorisables. (exemple : filtre sur le category=friend)

Etape 3 : Appel à des applications tiers (dépendance 1)

Certains champs (quand ils sont sélectionnés, peuvent être associés à des commandes particulières (autre que Full Display, Edit). Par exemple : le champ phone ajoute la commande Call qui en principe appèle l'application de téléphonie. Le champ gsm ajoute la commande Call et la commande SMS qui en principe appèle l'application de rédaction et d'envoie de SMS. Le champ url ajoute la commande Browse qui en principe appèle le navigateur.
Vous n'effectuerez l'appel réel aux applications-tierces.

Etape 4 : La sécurisation des entrées (dépendance 1)

Cette étape consiste à protéger les entrées du carnet par un mot de passe. Vous utiliserez pour cela des moyens cryptographiques (voir exemple crypto)
Vous pourrez étendre la classe RecordStore par une classe SecureRecordStore qui s'initialise avec un encrypteur et un décrypteur générique. Utilisez l'instanciation retardée de l'encrypteur/decrypteur.

Etape 5 : Le déchargement (ou export) (dépendance 1)

Cette étape consiste à exporter le contenu des entrées du carnet vers une servlet. Vous definirez une DTD XML pour le format du document à exporter. Vous utilisez la méthode POST pour envoyer ce document à la servlet AddressBookServlet. Ecrivez la servlet AddressBookServlet. Remarque : la servlet gère plusieurs carnets d'adresses. Le carnet est sélectionné après authentification simple.

Etape 6 : Le chargement (ou import) (dépendance 1)

Cette étape consiste à importer des entrées dans le carnet, chargées depuis une servlet. Vous définirez une DTD XML pour le format du document à importer (il peut être identique à celui de l'import). Comme il peut y avoir des conflits avec les entrées déjà présentes dans le carnet, définissez les champs qui forment la clé des entrées et proposez plusieurs politiques d'importation (Remise à zéro, Remplacement, Fusion, Abandon de l'entrée). Vous utilisez la méthode GET pour recevoir ce document à la servlet AddressBookServlet. Completez la servlet AddressBookServlet.

Etape 7 : La synchronisation (dépendance 5 et 6)

Cette étape consiste à synchroniser les entrées dans le carnet et celui géré par la servlet AddressBookServlet. Vous pourrez vous inspirer de SyncML (http://www.syncml.org) et lire « Sync or Sink--Synchronizing Data Between Clients Based on the Java™ 2 Platform, Micro Edition (J2ME™) and Servers Based on the Java 2 Platform, Enterprise Edition (J2EE™) " http://servlet.java.sun.com/javaone/resources/content/sf2002/conf/sessions/pdfs/1812.pdf. et utiliser kSync (http://ksync.enhydra.org). Complétez la servlet AddressBookServlet. Utilisez un drapeau de synchronisation sur les entrées indiquant que ces dernières ont été modifiées depuis la dernière synchronisation. Définissez les politiques de règlements de conflits qui seront appliqués bilatéralement.