MCA PART II

Paper-XI

Topic: MOBILE SOFTWARE ENGINEERING

Prepared by: Dr. Kiran Pandey

(School of Computer Science)

Email-id: kiranpandey.nou@gmail.com

INTRODUCTION

With the increased usage of mobile devices across the world, more and more applications are being targeted at them. If an application that is developed for computers can also run on the mobile devices, then the entire paradigm of application usage changes as it enables the user to run application from wherever he is.

There were different standards on which the mobile networks are based. One of the popular standard is GSM. In this unit, we introduce the GSM architecture. There were a number of environments under which mobile applications can be developed. One of the popular environments is Java. We introduce J2ME (Java 2 Micro Edition) in this unit in conjunction with wireless application development with it. Any application should be tested before being put to full-fledged use. In this unit, Java Device Test Suite is introduced with the help of which wireless applications can be tested.

INTRODUCTION TO GSM

Mobile devices are used by a number of people across many countries. In this unit, mobile devices refer to mobile phones. Every country is having a set of standards on which the mobile devices in their country are based upon. GSM is one of the popular architectures on which mobile devices are based on. GSM stands for Global System for Mobile Communications.

GSM is a digital wireless network standard. All mobile devices that are based on GSM standard across the world will have similar capabilities.

The following are some of the features of GSM:

- If a mobile device is based on GSM, then it can be used in all those countries where this particular standard is prevailing.

- Almost all the services that are existent in a wireline network are provided by GSM to all the users of mobile devices which are based on it.

- Though the quality of voice telephony is not excellent, it is not inferior to the systems that are analog based.

- It also provides good security as there is an option to encrypt the information that is being exchanged using this standard.

- There is no need for significant modification of wireline networks due to the establishment of networks based on GSM standard.

Architecture of GSM

There are three different parts in a mobile device. They are SIM card, Mobile equipment and Terminal equipment. SIM stands for Subscriber Identity Module. A mobile device can be loaded by any SIM. The number of the mobile device is associated with the SIM. The size of a SIM is similar to that of a smart card. Every SIM is also having a PIN (Personal Identity Number) associated with it. After loading the SIM into the mobile device, the user is asked to enter the PIN. Only on entering the correct PIN, the user will be able to start using the services offered by the mobile operator. Initially, PIN is loaded by the operator. The PIN can be changed by the user. If the user is unable to enter the SIM correctly for the first time, then s/he is given an opportunity to re-enter it for a fixed number of times. In case, s/he is not able to enter it correctly and exhausted all the tries, then the PIN is blocked. So, that particular SIM cannot be used unless the network operator activates it. The SIM can be unblocked only on entering the correct PUK (PIN Unblocking Key). Information related to subscriber, PIN and PUK codes are present in SIM.

The architecture of GSM is composed of Mobile devices, Base Station Systems (BSS), Mobile Switching Center (MSC) etc. The communication between Mobile device and BSS is through radio interface. BSS communicates with MSC by connecting to Network and Switching Subsystem (NSS). An MSC is a telephone exchange that is specifically configured for mobile applications. Mobile devices and Public Switched Telephone Network (PSTN) interface through Base stations. BSS consists of a Base Transceiver Station (BTS) and Base Station Controller (BSC). Equipment related to transmission, reception and signalling is part of BTS and this equipment is used by it to contact Mobile devices. BSC deals with the allocation of radio channel and its release apart from hand off management. Using ISDN protocols, BSC communicates with BTS.

Apart from GSM, there are other standards such as CDMA etc.

WIRELESS APPLICATION DEVELOPMENT

USING J2ME

Java supports mobile application development using its J2ME. J2ME stands for Java 2 Platform, Micro Edition. J2ME provides an environment under which application development can be done for mobile phone, personal digital assistants and other embedded devices. As like any other Java platform, J2ME includes API's (Application Programming Interface) and Java Virtual Machines. It includes a range of user interfaces, provides security and supports a large number of network protocols. J2ME also supports the concept of write once, run anywhere concept. Initially, an application can be developed using J2ME targeting a specific type of devices. Then, the same application can be used for different types of devices. Also, it is possible to use the native capabilities of these devices. J2ME is one of the popular platforms that is being used across the world for a number of mobile devices, embedded devices etc. There is a huge market for applications that target wireless world. Some of the wireless applications are Games, Applications that access databases, Location based services etc.

The architecture of J2ME consists of a number of components that can be used to construct a suitable Java Runtime Environment (JRE) for a set of mobile devices. When right components are selected, it will lead to a good memory, processing strength and I/O capabilities for the set of devices for which JRE is being constructed.

Currently, two configurations are supported by J2ME. They are Connected Limited Device Configuration (CLDC) and Connected Device Configuration (CDC). Each configuration consists of a virtual machine and a set of class libraries. Each configuration provides functionality of a particular level to the set of devices that have similar features. One of the features is network connectivity. To provide a full set of functionalities, a profile and an additional set of APIs should be added to the configuration. These additional APIs will enable the usage of native properties of devices as well as define suitable user interface. Each profile will support a smaller set of devices among the devices that are supported by the chosen configuration.

In the case of J2ME, often CLDC is combined with MIDP. MIDP stands for Mobile Information Device Profile. This combination provides a Java based environment for cell phones in which applications can be developed. It is always possible to add optional packages to the configuration that was already selected to support an additional set of services. An optional package consists of a set of standard APIs that help us to use latest technologies such as connecting to databases from cell phones, blue tooth, multimedia etc. It is never mandatory to use optional packages. In fact, it is to the developer to select those optional packages which provide the additional functionality as per his desire.

CDC is the larger of the two configurations supported by J2ME. CLDC is designed for devices that have less memory, slow processor and unstable network connections. CLDC is suitable for cell phones. Such devices have a memory that ranges from 128KB to 512KB. Java has to be implemented within this memory. CDC is designed for devices that have larger memory, faster processors and a significant network bandwidth. TV set top boxes, high range PDAs (Personal Digital Assistant) are examples of the devices that are suitable for CDC. CDC consists of a JVM (Java

Virtual Machine) and Java software that includes a significant portion of J2SE (Java 2 Standard Edition) platform. Usually, devices that are suitable CDC at least possess 2MB of memory in which Java software can be loaded.

MIDP is designed for cell phones and other lower level PDAs. User interface, network connectivity, local data storage and application management are offered by MIDP for applications that are based on mobile devices. The combination of MIDP and CLDC provides a full-fledged Java runtime environment for the mobile devices. The environment provided by their combination will add to the features of the mobile devices. This will lead to a better power consumption and efficiency in memory utilisation. The lowest level profile for CDC is known as Foundation Profile (FP). FP provides embedded devices which does not have a user interface with network capability. There are two more profiles, namely, Personal Basis Profile (PBP) and Personal Profile (PP). FP can be combined with these profiles to provide GUI (Graphical User Interface) for the devices that require it. All the profiles of CDC are in a layered manner. This will facilitate adding as well as removing profiles as per the need of features.

PP is useful for all the devices that require a GUI, Internet support etc. Usually, devices such as higher level PDAs, Communicators etc. need such features. Java AWT (Abstract Window Toolkit) is contained in this profile. It also offers web fidelity, applets etc. PBP is a subset of PP. PBP supports application environment to those devices in the network that support at least a basic graphical representation. PP as well as PBP are layered to the top of CDC and FP.

There are several reasons for the usage of Java technology for wireless application development. Some of them are given below:

- Java platform is secure. It is safe. It always works within the boundaries of Java Virtual Machine. Hence, if something goes wrong, then only JVM is corrupted. The device is never damaged.

- Automatic garbage collection is provided by Java. In the absence of automatic garbage collection, it is to the developer to search for the memory leaks.

- Java offers exception handling mechanism. Such a mechanism facilitates the creation of robust applications.

- Java is portable. Suppose that you develop an application using MIDP. This application can be executed on any mobile device that implements MIDP specification. Due to the feature of portability, it is possible to move applications to specific devices over the air.

You can use J2ME wireless toolkit for development of wireless applications using J2ME. It is possible to set up a development environment by using the following software:

- J2SE (Java 2 Standard Edition) SDK (Software Development Kit) of v1.4.2 or upwards.
- J2ME (Java 2 Micro Edition) Wireless toolkit. Using this toolkit, MIDlets can be developed.
- Any Text editor.

MIDlets are programs that use the Mobile Information Device Profile (MIDP).

The first step is to install J2SE SDK. J2ME wireless tool kit runs on the Java platform provided by J2SE. J2SE needs to be installed because it consists of Java compiler as well as other requisite software which is used by J2ME wireless tool kit to build programs.

The second step is to install J2ME wireless toolkit. It is possible to develop MIDP applications using J2ME wireless toolkit. The J2ME wireless toolkit uses the concept of projects rather than files. Once J2ME wireless toolkit is installed and run, we can start creating projects. At the end of the creation of the project, you have on MIDP suite. Once the project is created, its properties can be changed. Also, the project can be build and executed in the device emulator. Let us create a new project. Let the title of the project be project1. Along with the title of the project, MIDlet class name should also be given. Let it be projectlet1.

Once these names are confirmed, the process of creating the project project1 commences. As the result of creation, a directory titled project1 will be created in which the following subdirectories will be present:

- bin
- lib
- res
- src

A compiled MIDlet suite (a .jar file) and the MIDlet suite descriptor (a .jad file) are created in the bin directory. Extra JAR that are to be used in the project may be placed in the lib directory. Any resource files such as images, text files etc. may be placed in the RES directory. The source code created by you will be in SRC directory. Apart from the above mentioned subdirectories, the following subdirectories are also created:

- tmpclasses

- tmplib

But, these subdirectories are not used by the developer explicitly. They are basically used internally.

The details of code with which MIDlets can be developed is a larger topic and references may be studied for it.

Once the MIDlet projectlet1 is developed, it should be saved as projectlet1.java file in the SRC directory. The following is the exact location of this file:

- D:\WTK22\apps\project1\src\projectlet1.java

Now, build the project. After successful build, run it. Upon running, a mobile phone emulator will pop up. In the emulator, the title of the project will be displayed along with a button labelled Launch. On clicking it, the MIDlet developed will be executed and the result is displayed on the screen. There is a button labelled Exit at the bottom of the display on which the MIDlet is executed. If the Exit button is clicked, then, we exit the MIDlet that is executed. Now, there are two ways to exit the emulator. Either the emulator window can be closed or ESC key can be pressed.

There are a number of emulators provided by J2ME wireless toolkit. One of them can be selected from the KT (ToolKit) tool bar. After selection, we need to execute the project again.

When the project is build, the toolkit will compile all the .java files in the src directory. The compilation is performed in the MIDP environment. There were different APIs in the MIDP and J2SE environment. When the project is being compiled in the MIDP environment, then the APIs from MIDP should be considered for the classes that are used in MIDlets. The corresponding APIs in J2SE are not used. All MIDP classes are verified in two phases before they are loaded and executed. At build time, the first verification is done by J2ME wireless toolkit. When classes are loaded, the second verification is done by device's runtime system. Finally, all MIDlets are packaged into MIDlet suites. These suites are distributed to actual devices.

Along with the J2ME wireless toolkit, information about the following is also provided:

- Application development cycle
- Attributes of MIDlet
- Files in the installed directories
- Device types
- Portability

- Guidelines on the process of configuring an emulator
- Usage of J2ME wireless toolkit.

Until now, we focused on the development of MIDlet. It is possible to do a bit of extra work and make these MIDlets work across networks. For this to happen, we may need to develop servlets to whom MIDlets can establish network connections.

For developing servlets, we need server. Tomcat is the server that can be used for this purpose. There are a number of servers available and every server is having its own advantages and disadvantages.

The first step is to install and run Tomcat. The latest version can be downloaded from the website of Apache Jakarta Project. It comes as a ZIP archive. After downloading, it can be installed into any directory. Tomcat is written in Java. Tomcat needs the location of J2SE to run. Hence, the location of J2SE should be set in the JAVA_HOME environment variable. Once, all these steps are performed, then, open the command window. Change to the bin directory of Tomcat. Type startup at the prompt. That's it. Tomcat starts running. Don't worry about the things that are displayed on the screen once you start it.

It is possible to check whether Tomcat is really running or not. For that, just open the browser window. Try to open http://localhost:8080/. If a default page from Tomcat with links to servlet and JSP examples is displayed, then, it means that the Tomcat is running. To shutdown the Tomcat, open the command window. Change to bin directory of Tomcat and then type the command shutdown. That's it. Tomcat starts shutting down.

Now, suppose that a servlet called counter is developed. The name of the file will be counter.java. It should be saved to the root directory of Tomcat. Once it is saved to the root directory, it can be compiled. To compile servlets, there is need for servlet API. This should be added to the CLASSPATH before compiling servlets. The servlet API is present in common/lib/servlet.jar under the Tomcat directory. Now, the counter.java servlet can be compiled using javac as follows:

D:\>javac counter.java

Let us place this servlet in a web application. The servlet can be saved to a directory in webapps directory of Tomcat. Let this directory be abcd. Tomcat will know about the new web application through its configuration files. The necessary changes to the configuration files needs to be done by the developer. This can be done by opening the server.xml file from conf directory and adding a context entry.

The reason for performing all these steps is to handle the incoming http requests. If the request begins with /abcd, then the request will be sent to the web application that is located at webapps/abcd. The most important file in any web application is web.xml. It is always stored in WEB-INF directory. This is the time when we start

thinking about making the servlet counter.java accessible to the entire world. Suppose that counter.java is placed in a directory called count.

Now, the following is the full path to the servlet:

http://localhost:8080/abcd/counts

Now, necessary changes are to be done to web.xml file so that Tomcat will know that counter servlet should be mapped to abcd/counts.

As we have seen, both servlets and MIDlets can connect to the world via http. Hence, connecting MIDlet to servlet is not a complicated issue.

The following are different steps in the process of connecting MIDlet to a servlet:

- Start Ktoolbar. This is part of J2ME wireless toolkit.
- Open the project1.
- Write the code for the MIDlet (let it be countmidlet.java) that connects to counter servlet.
- The screen of countmidlet.java (after executing it) consists of two commands namely Exit and Connect. Clicking on Connect will lead to the invocation of connect ( ) method that will establish a network connection. It will also transport the result.
- The countmidlet.java should be saved to apps/project1/src directory under J2ME wireless toolkit directory.
- Now, J2ME wireless toolkit should know that a new MIDlet is added to it. This can be done by going to Settings->MIDlets->Add. Enter countmidlet for both the MIDlet name and class name. Click on OK.
- Go to Settings->User Defined. Add the property name as countmidlet.URL. This URL will invoke counter servlet.