Java in Wireless 개발자를 위한 지침서(4)

End to End Java 애플리케이션 구축하기(1)

정리 · 김경한 | 한국 썬 시스템엔지니어링 본부 대리

이번 호부터 2회에 걸쳐 Java™ 2 Platform, Micro Edition(J2ME™) 환경에서 End to End Java 애플리케이션을 개발하고 실행하는 데 필요한 A에서 Z까지 자세하게 살펴볼 것이다. 여기서는 개발 툴을 설치하고 J2ME 애플리케이션을 코딩 및 빌드하며 에뮬레이터에서 테스트할 수 있는 방법 등에 대해 알아본다.

난이도 상 중 하

여기서 만들어 볼 애플리케이션인 MIDlet은 J2ME 규격 중 하나인 Mobile Information Device Profile의 구현 상에서 실행이 될 것이다. 참고로 'http://wireless.java.sun.com/getstart/'에서 무선 Java 기술에 대한 배경 지식을 얻을 수 있다.

■ 개발 환경 구축을 위한 요구 사항

MIDP 개발 환경의 시스템 요구 사항은 간단한 선택에 의해 결정된다. 즉 Sun™ ONE Studio Mobile Edition의 사용 유무이다. 이것은 무료 통합 개발 환경(IDE)으로 애플리케이션을 개발하는 데에 많은 작업을 자동으로 처리해준다. 단 Sun™ ONE Studio를 사용하면 시스템의 프로세서와 메모리 및 하드디스크 용량에 대한 요구 사항이 높아진다.

요구되는 시스템 사양은 다음과 같다.

- 166MHz 또는 그 이상(Sun™ ONE Studio를 사용하면 300MHz 또는 그 이상)
- 64MB의 메모리(Sun™ ONE Studio를 사용하면 128MB)
- 30MB의 하드디스크 용량(Sun™ ONE Studio를 사용하면 140MB)

MIDP 개발 툴은 Linux와 Solaris™, Windows 운영 체계에서 사용할 수 있다. 개발 환경은 다음의 세 가지 소프트웨어로 구성돼 있다.

- Java™ 2 Standard Edition(J2SE™) SDK 1.3 버전 이상
- J2ME Wireless Toolkit(J2MEWTK). 이 툴의 패키지는 MIDlet을 만들고 테스트하기 위한 것이다

- 텍스트 에디터. 이것은 Windows에 있는 노트패드와 같이 간단한 것이지만 보다 정교한 jEdit(http://www.jedit.org)이 있다

어떤 에디터를 사용할 것인지는 완전히 개발자들의 마음이다. Unix 시스템에서는 emacs 또는 vi가 인기있다. 어떤 Windows 개발자들은 노트패드를 사용하지만 좀더 복잡한 개발이라면 이것보다 좀더 고급 기능이 있는 것을 원할 것이다. jEdit은 Java 2 런타임에서 실행되며 Windows 2000이나 맥 OS X과 같은 다른 시스템에서도 잘 동작한다. Sun™ ONE Studio는 고유의 에디터를 가지고 있다.

J2MEWTK는 독립적으로 실행될 수도 있고 Sun™ ONE Studio의 일부로 실행될 수도 있다.

J2SE SDK 설치하기

Sun™ ONE Studio의 사용 유무에 관계없이 J2SE SDK는 필요하다(간혹 다른 개발자들이 이것을 JDK 또는 Java Developer's Kit이라고 부르기도 하지만, 현재 명칭은 J2SE SDK이다). 이것은 'http://java.sun.com/j2se/'에서 최신 버전인 J2SE SDK v1.3 또는 1.4를 다운받을 수 있으며, Linux와 Solaris, Windows에서 사용될 수 있다.

그럼 J2SE SDK가 무선 애플리케이션을 개발하는 데 필요한 이유는 무엇일까? 그 이유는 J2ME Wireless Toolkit이 동작할 수 있는 Java 플랫폼과 Java 컴파일러, J2MEWTK가 프로젝트를 빌드하는 데 사용하는 툴들을 제공하기 때문이다.

J2SE를 다운받았으면 설치하도록 하자. Windows에서 다운받은 파일을 실행시키자. 설치 프로그램은 몇 가지 질문을 하고 소프트웨어를 설치할 것이다. 기본 설정으로 설치했으면 J2SE는 'c:\jdk1.3.1' 또는 'c:\jdk1.3.1_01'에 있을 것이다. 그 아래에 있는 bin 디렉토리를 autoexec.bat 파일(Windows 95/98) 또는 시스템 정보(Windows NT/2000)를 통해 경로를 추가해야 한다. autoexec.bat 파일의 발췌문을 보면 J2SE SDK의 bin 디렉토리를 Windows 98 경로에 추가하는 방법을 알 수 있을 것이다.

```
path %path%;c:\jdk1.3.1\bin
```

Windows NT/2000의 경우 제어판/시스템/고급탭/환경 변수 메뉴를 이용하라

제대로 설치되었는지 테스트하려면 커맨드 프롬프트를 연다(autoexec.bat를 변경했다면 먼저 재시동해야 한다). 여기에서 java –version을 입력하면 다음과 같은 내용이 출력된다.

```
C:\>java -version
java version "1.3.1_01"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.3.1_01)
Java HotSpot(TM) Client VM (build 1.3.1_01, mixed mode)

C:\>
```

J2ME Wireless Toolkit 설치하기

다음으로 할 일은 J2MEWTK을 설치하는 것이다. 이것은 MIDP 애플리케이션을 쉽게 빌드하고 테스트할 수 있도록 도와주는 툴 셋이다. 참고로 J2MEWTK는 IDE의 축소판이라고 볼 수 있으며, MIDP 애플리케이션을 빌드하는 데 여러 작업을 자동으로 처리해준다.

J2MEWTK는 'http://java.sun.com/products/j2mewtoolkit/'에서 다운받으면 된다. 그런 다음 설치 파일을 실행하면, 설치 프로그램은 J2SE SDK를 검색할 것이다. 만약 문제가 있다면 J2SE SDK가 설치된 디렉토리에 J2MEWTK를 독립적으로(독립 모드) 실행할 것인지, Sun™ ONE Studio에 통합해 실행할 것인지를 설정해준다. 여기서는 J2MEWTK를 독립 모드로 실행한다고 가정하자. J2MEWTK 파일은 사용자가 특별히 별도의 디렉토리를 설정해주지 않는 한 c:\J2MEWTK에 설치되고, 설치 프로그램은 툴킷의 여러 부분에 대한 바로가기를 생성할 것이다.

툴킷을 실행하기 위해 KToolbar 바로가기를 실행하면 그림 1과 같은 화면이 나타날 것이다.

J2MEWTK는 projects 단위로 작업한다. 하나의 프로젝트는 하나의 MIDlet suite이다. 툴킷은 한 번에 하나의 프로젝트만 작업할 수 있다. 개발자는 현재 프로젝트의 속성을 변경시키고, 프로젝트를 빌드하며, 기기 에뮬레이터에서 프로젝트를 실행할 수 있다. 툴킷을 설치할 때 여러 개의 예제 프로젝트도 설치된다. 이것은 추후에 살펴보도록 한다.

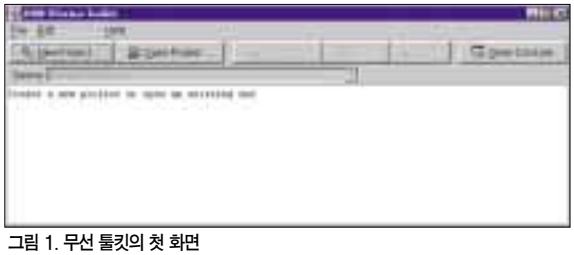

그림 1. 무선 툴킷의 첫 화면

그림 2. 새로운 프로젝트 생성하기

새 프로젝트를 생성해 바로 시작하자. 먼저 'New Project' 버튼을 누르면 J2MEWTK가 프로젝트명과 MIDlet 클래스명을 물을 것이다. 그림 2와 같이 입력하자.

프로젝트명과 MIDlet명을 입력하면 J2MEWTK는 프로젝트 설정을 변경할 기회를 준다. 현재로선 기본 설정을 그대로 사용하고 'OK' 버튼을 눌러서 새 프로젝트를 생성한다. J2MEWTK의 출력창을 보면 프로젝트에 사용할 소스 파일들을 어디에 넣을지 알려주는 메시지가 나타난다. 다음과 같은 메시지가 출력되었다.

```
Creating project "HelloSuite"
Place Java source files in "c:\J2MEWTK\apps\HelloSuite\src"
Place Application resource files in "c:\J2MEWTK\apps\HelloSuite\res"
Place Application library files in "c:\J2MEWTK\apps\HelloSuite\lib"
```

J2MEWTK는 각 프로젝트를 apps 디렉토리 아래에 저장한다. 생성되는 디렉토리의 이름은 프로젝트명과 동일하다. 그림 3에서는 J2MEWTK가 새로운 디렉토리 c:\J2MEWTK\apps\HelloSuite를 생성한 것을 알 수 있다. 각 프로젝트의 하위 디렉토리 구조는 표준 구조를 갖는다.

bin 디렉토리에는 컴파일된 MIDlet suite(.jar 파일)과 MIDlet suite 디스크립터(.jad 파일)가 포함된다. lib 디렉토리는 프로젝트에서 추가적으로 사용할 모든 JAR 파일이 위치하는 곳이다. res 디렉토리는 MIDlet suite에 포함될 그림이나 텍스트 파일과 같은 리소스 파일들이 포함된다. 마지막으로 src 디렉토리는 소스 코드를 저장하는 곳이다. 패키지와 디렉토리의 표준 규칙이 적용된다. 예를 들면 users의 Root 클래스 소스 파일은 src/users/Root.java에 들어간다.

KToolbar를 사용해 프로젝트를 생성하면 몇 개의 디렉토리가 추가적으로 생성된다. 그림 4에서 보는 바와 같이 J2MEWTK는 classes와 tmpclasses, tmplib를 추가적으로 생성했다. 그러나 이런 부분은 J2MEWTK가 내부적으로만 사용하기 때문에 신경쓰지 않아도 된다.

그림 3. 프로젝트 디렉토리 구조

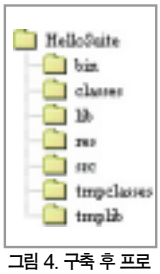

그림 4. 구축 후 프로젝트 디렉토리 구조

MIDlet 만들기

MIDlet을 개발하기 위해 간단한 MIDlet을 작성해보자. 선택한 텍스트 에디터에 다음의 코드를 타이핑하자.

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

public class HelloMIDlet
  extends MIDlet
  implements CommandListener {
  private Form mMainForm;

  public HelloMIDlet() {
    mMainForm = new Form("HelloMIDlet");
    mMainForm.append(new StringItem(null, "Hello, MIDP!"));
    mMainForm.addCommand(new Command("Exit", Command.EXIT, 0));
    mMainForm.setCommandListener(this);
  }

  public void startApp() {
    Display.getDisplay(this).setCurrent(mMainForm);
  }

  public void pauseApp() {}

  public void destroyApp(boolean unconditional) {}

  public void commandAction(Command c, Displayable s) {
    notifyDestroyed();
  }
}
```

이 코드를 HelloMIDlet.java로 프로젝트의 src 디렉토리에 저장하자. 여기에서는 c:\J2mewtk\apps\HelloSuite\src\HelloMIDlet.java에 파일이 저장된다. KToolbar에 있는 'Build' 버튼을 누르자. J2MEWTK는 프로젝트를 컴파일한다. 만약 컴파

일 에러가 발생하면 KToolbar의 출력 창에 나타날 것이다.

이제 MIDlet suite를 테스트할 준비가 되었다. 'Run' 버튼을 누르면 그림 5와 같이 모바일 전화기 에뮬레이터가 나타난다.

그림 5. 에뮬레이터에서 HelloSuite을 실행하고 있다

에뮬레이터에서 MIDlet suite를 보면 MIDlet의 목록을 보여준다. 이 예제에서는 하나의 MIDlet만 보여주고 있다. HelloSuite란 이름을 가지고 있지만 실제 실행될 클래스는 HelloMIDlet이다. 어떻게 연결돼 있는지 보려면 KToolbar에서 'Settings...'을 누르고 'MIDlets' 탭을 누르면 프로젝트의 MIDlet 목록이 나타날 것이다.

에뮬레이터로 돌아와서 화면 안의 'Launch' 버튼을 눌러 MIDlet을 실행하자. 그림 6과 같이 간단한 화면이 나타난다. 그런 다음 'Exit'을 눌러 MIDlet에서 나오자. 에뮬레이터 창을 닫거나 ESC 키를 눌러 에뮬레이터 세션을 끝내자.

그림 6. 에뮬레이터로 HelloMIDlet을 실행하고 있다

여기서 사용한 에뮬레이터는 DefaultGrayPhone이다. J2MEWTK는 이밖에도 호출기나 컬러폰과 모토로라 i85s 전화기, RIM 블랙베리와 같은 실제 기기의 에뮬레이터도 가지고 있다. HelloMIDlet을 다른 기기 에뮬레이터에서도 실행해 사용자 인터페이스가 어떻게 적용되는지 시험해보자(Palm OS를 위한 MIDP 에뮬레이터는 Palm OS 에뮬레이터, 즉 POSE를 'http://www.palmos.com/dev/'에서 별도로 다운받아서 설치해야 한다). KToolbar에서 원하는 에뮬레이터를 선택하고 'Run' 버튼을 누르기만 하면 된다.

HelloMIDlet을 충분히 살펴보았다면 J2MEWTK에 번들되어 있는 다른 프로젝트를 시도해보는 것도 좋을 것같다. 데모나 게임, 포토 앨범 샘플 프로젝트가 있으니 한번 시도해보기 바란다.

■ 작동 원리 이해하기

재미있게 MIDlet을 만들어봤으니, 다시 처음으로 돌아가 J2MEWTK의 역할이 무엇인가를 알아보자. 그리 복잡한 것은 없지만 J2MEWTK는 번거로울 수 있는 여러 단계를 간략하게 하나의 버튼 클릭으로 만들었다.

처음에 'Build' 버튼을 눌렀을 때 어떤 일이 일어났는가? J2MEWTK는 해당 프로젝트의 src 디렉토리에 있는 모든 .java 파일을 찾아 컴파일한다. 소스 파일은 J2SE 환경이 아닌 MIDP 환경에 맞게 컴파일돼야 하기 때문에 일반 컴파일과 다르다고 볼 수 있다. 이 차이점을 이해하기 위해서는 java.lang.System을 사용하는 MIDlet을 생각하면 된다. J2SE에 있는 java.lang.System과 MIDP에 있는 클래스는 서로 다른 API를 가지고 있다. 따라서 J2MEWTK가 MIDlet 클래스를 컴파일할 때, J2SE 버전이 아닌 MIDP의 java.lang.System을 사용해야 한다.

어떠한 클래스를 사용할 것인가는 javac에 -bootclasspath이라는 옵션을 사용해 지정할 수 있지만, J2MEWTK가 알아서 처리하도록 하는 것이 편하다.

컴파일한 후 MIDP 클래스는 MIDP 기기에서 실행되기 전에 preverified가 되어야 한다. J2SE에 .class 파일을 로드하기 전에 바이트코드 검증기가 있다는 것을 기억할 것이다. MIDP에서는 검증을 두 단계로 나눈다. 툴킷이 빌드할 때 초기 검증하고 클래스가 로드할 때 기기의 런타임 시스템에서 두 번째 검증을 행한다.

커맨드 라인에서 사용되는 preverify 툴을 사용해서 첫 번째 검증을 해볼 수 있지만 자세한 것은 J2MEWTK에 맡겨두는 것이 더 쉽다.

마지막으로 MIDlet들은 기기에 배포될 MIDlet suites에 포함된다. 이 과정에서 MIDlet suites 클래스 파일과 리소스 파일을 JAR 파일로 만들고, JAR manifest에 정보를 추가하는 일이 수반된다. 이런 절차는 J2MEWTK에 맡기는 것이 최상의 방법이다. MIDlet suite을 만들려면 메뉴에서 'Project | Package'를 선택한다. MIDlet suite용으로 .jad와 .jar 파일이 생성되어 프로젝트의 bin 디렉토리에 저장된다.

■ 마치며

이제 MIDP 개발에 대한 기본적인 것을 이해하고 MIDlet suite을 만드는 데 필요한 소프트웨어를 가진 셈이다. 이것만으로도 주목할 만한 일이다. 그러나 MIDP 클라이언트 프로그래밍은 반쪽짜리일 뿐이다. 특수한 경우를 제외하고는 대부분의 MIDlet은 네트워 서비스에 연결하게 될 것이다. 따라서 다음 호에는 서버 환경을 어떻게 설치 및 설정하고 실행하는지 알아보고, 간단한 Servlet을 만든 다음 MIDlet을 약간 수정해 Servlet에 네트워을 연결하는 방법을 살펴본다.