Wireless Java Technology

Pao-Ann Hsiung

National Chung Cheng University

Ref: http://developers.sun.com/techtopics/mobility/learning/tutorial/

Contents

- Overview of Java 2 Platform
- Overview of J2ME
- Scope of Wireless Java Technology
- Wireless Development Tutorial Part I
- Wireless Development Tutorial Part II

Overview of Java 2 Platform

- Java Programming Language
  - Type safe object references (instead of pointers)
  - Automatic garbage collection
  - Interfaces (instead of multiple inheritance)
- Virtual Machine
  - Runs on various OS and HW
  - Safe execution of untrusted code
- APIs
  - From UI to cryptography,
  - from CORBA to internalization

Overview of Java 2 Platform

- Java 2, Standard Edition (J2SE)
  - Desktop computers, OS X, Linux, Solaris, Windows
- Java 2, Enterprise Edition (J2EE)
  - Multiuser, enterprise-wide applications
  - J2SE + Server-side computing
- Java 2, Micro Edition (J2ME)
  - Small devices: mobile phones, set-top boxes
  - Smaller VM, leaner APIs than J2SE

Overview of J2ME

- J2ME is a platform of:
  - Configurations
  - Profiles
  - Optional packages

Configurations

- Virtual Machine Specifications
  - Full JVM or a subset
- Base set of API for a certain class of device
  - Subset of J2SE APIs
- Example
  - for devices with <512 KB memory, intermittent network connection
- Implementations
  - CLDC, CDC

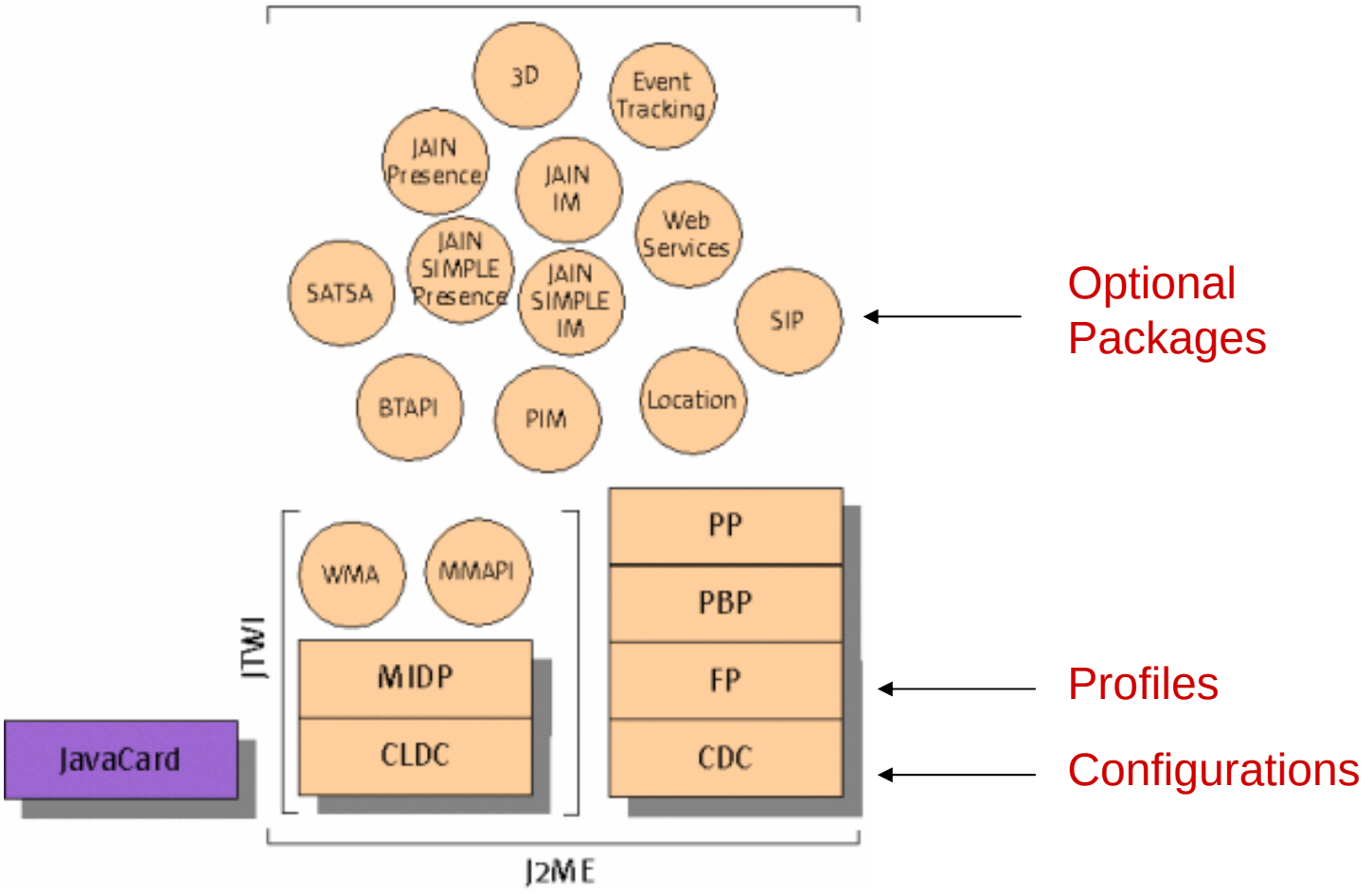
Profiles

- Profile consists of
  - Configuration
  - Application-specific APIs
  - User Interface
  - Persistent Storage
- Examples
  - Mobile Information Device Profile (MIDP)
  - Foundation Profile (FP)
  - Personal Basis Profile (PBP)
  - Personal Profile (PP)

Optional Packages

- Functionality not associated with a specific configuration or profile
- Example
  - Bluetooth API (BTAPI, JSR 82)
  - Web Services, Mobile Media API (MMAPI), Wireless Messaging API (WMA), Security and Trust Services API (SATSA), Location API, Content Handling API (CHAPI), Event Tracking, Mobile 3D Graphics API, Scalable 2D Vector Graphics API, Payment API, Data Sync API, …

The J2ME Universe Today

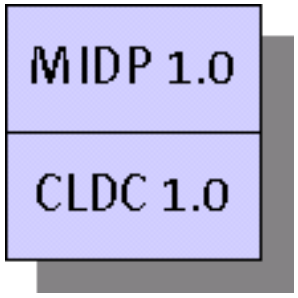


The J2ME Universe Today

- Two Main Branches
  - Connected, Limited Device Configuration (CLDC)
    - Small wireless devices with intermittent network
    - Examples: Pagers, mobile phones, PDAs
  - Connected Device Configuration (CDC)
    - Larger devices (in terms of memory and processing power)
    - Examples: set-top boxes, internet appliances, high-end PDAs (Sharp Zaurus)

Software Stacks

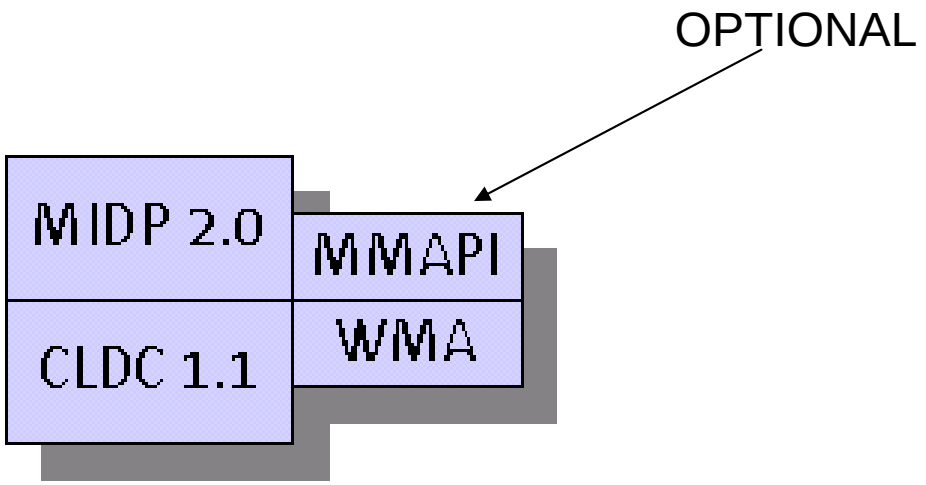
- A configuration
- A profile
- Some optional APIs
- Example 1: 1st Generation J2ME mobile phones



J2ME Stack

Software Stacks

- Example 2: JSR 185 mandates
  - CLDC 1.0 or 1.1
  - MIDP 2.0
  - WMA



JSR 185 Stack

Scope of Wireless Java Technology

- Intersection of two vast worlds
  - Wireless data communication
  - The Java Platform
- Misconceptions
  - Wireless Java Technology and J2ME are not same
    - CDC devices may have wired connections
    - Laptop running J2SE applications, connected by 802.11
  - MIDP is not all of J2ME
    - First finished profile
  - MIDP is not all of wireless technology
    - PP, J2SE on wireless devices, PDA profiles

Why Java for Wireless Development?

- Java platform is safe.
  - In JVM
- Java encourages robust programming.
  - Garbage collection, exception mechanism
- Portability
  - When you write a MIDlet (a MIDP application), it will run on any device that implements the MIDP specification
  - Over-the-air provisioning of applications

Wireless Development Tutorial Part I

- To create a Java Wireless Application called a MIDlet, that runs on implementations of the Mobile Information Device Profile (MIDP)
- MIDlets will connect to some type of network service
  - Part II will describe how to setup a servlet development environment, how to write, compile, and test a servlet

Software Required

- OS
  - Windows, Linux, OS X, Solaris
- Java 2 Standard Edition SDK 1.4.2 or higher
  - Currently 1.5.0 update 6
  - http://java.sun.com/j2se/
- Java Wireless Toolkit 2.2 or higher
  - Currently 2.3 Beta
  - http://java.sun.com/products/j2mewtoolkit/
- Text Editor
  - Notepad on Windows, jEdit, etc.

Development Environment

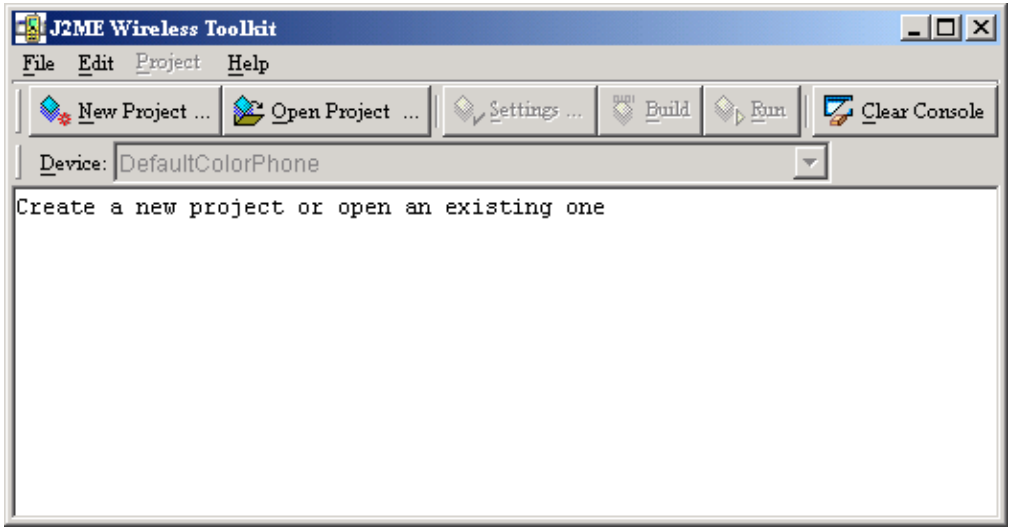
- Install J2SE SDK
  - Provides Java platform for Java Wireless Toolkit to run
  - Provides Java compiler and other tools to build your projects
  - To test installation, open “cmd” and type: “java –version”

```
Microsoft Windows XP [版本 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>java -version
java version "1.5.0_06"
Java(TM) 2 Runtime Environment, Standard Edition (build
  1.5.0_06-b05)
Java HotSpot(TM) Client VM (build 1.5.0_06-b05, mixed
  mode, sharing)
```

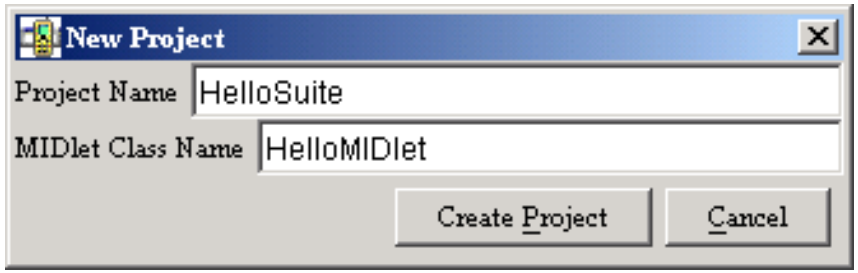
Development Environment

- Install Java Wireless Toolkit
  - To build and test MIDP applications
  - Will be installed in C:\WTK22
  - Run KToolbar
    - Works with projects
    - Result: MIDlet suite



Development Environment

- Create a new project



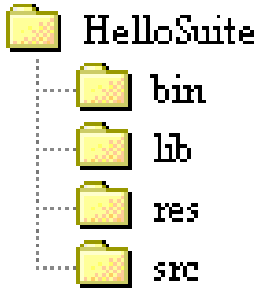
- Select OK for the default options

- Some helpful messages

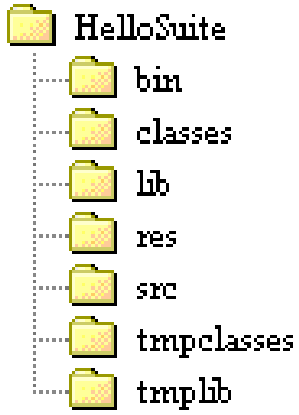
Creating project "HelloSuite"
Place Java source files in "C:\WTK22\apps\HelloSuite\src"
Place Application resource files in "C:\WTK22\apps\HelloSuite\res"
Place Application library files in "C:\WTK22\apps\HelloSuite\lib“

Development Environment

- Project is stored in C:\WTK22\apps\HelloSuite.
- Project Directory Structure:



- After building a project:



Creating a MIDlet

- Save the following code in the **`src`** directory with filename **`HelloMIDlet.java`**

```
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(); } }
```

You can also download from
http://developers.sun.com/techtopics/mobility/midp/articles/wtoolkit/src/HelloMIDlet.java

Creating a MIDlet

- Click “Build” button on the KToolbar
  - Debug if there are any errors
- Click “Run” button on the KToolbar
- The MIDlet is always run on a device emulator such as a Mobile Phone Emulator (see next slide)

Mobile Phone Emulator

- Click on the soft button under Launch

Mobile Phone Emulator

- The HelloSuite MIDlet runs in the emulator

Mobile Phone Emulator

- This is the DefaultColorPhone
- You can try other phone emulators in the combo box in the KToolbar
  - DefaultGrayPhone
  - MediaControlSkin
  - QwertyDevice

Looking under the hood

- What does "Build" do?
  - Compilation of .java files in **`src`** directory
    - Uses the MIDP API instead of the J2SE API
  - Preverification
    - Toolkit performs verification during build time
    - Device performs verification during load time
  - Bundling
    - JARing the MIDlet suite class files and the resource files
    - **Project | Package**
      - `.jad` and `.jar` files for MIDlet suite will be created in **`bin`**

Wireless Development Tutorial Part II

- To write and deploy a servlet
- Then, hook the MIDlet (from Part I) with the servlet
- Two versions of development platforms
  - Tomcat (currently 5.5.16 Beta)
    - Reference implementation for Java servlet and JSP
  - J2EE RI
    - Full implementation of J2EE specification

Installing and Running Tomcat

- Download from http://jakarta.apache.org/tomcat/
- Installs in C:\Program Files\Apache Software Foundation\Tomcat 5.5
- Start Tomcat (from the Start Menu)
- To check if Tomcat is running
  - http://localhost:8080/
  - A default page shows up: “TOMCAT_HOM”/webapps/ROOT/index.jsp

Writing Servlet Source Code

- Use text editor to create HitServlet.java in webapps/midp/WEB-INF/classes/ (you need to create midp and all subdirectories)

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class HitServlet extends HttpServlet {
    private int mCount;
    public void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
            String message = "Hits: " + ++mCount;
            response.setContentType("text/plain");
            response.setContentLength(message.length());
            PrintWriter out = response.getWriter();
            out.println(message);
    }
}
```

You can copy the code from http://developers.sun.com/techtopics/mobility/midp/articles/tutorial2/

Compiling the Servlet

- You need to set an environment variable
- C:\>set CLASSPATH=\Program Files\Apache Software Foundation\Tomcat 5.5\common\lib\servlet-api.jar
- Compile the Servlet in a command window:

  C:\Program Files\Java\jdk1.5.0_06\bin>javac "\Program Files\Apache Software Foundation\Tomcat 5.5\webapps\midp\WEB-INF\classes\HitServlet.java“
- It will generate a HitServlet.class file.

Deploying the Servlet

- Web Application
  - HTML, image files, servlets, other resources
  - Accessible via a web interface
- Tomcat has several web applications installed in webapps directory
  - examples, webdav
- Let’s create a web application and place our servlet inside

Deploying the Servlet

- To tell Tomcat about our new web application
  - Select Tomcat Manager from the Start menu
  - Enter the admin username and password, which you entered when installing Tomcat
  - The Tomcat Manager starts up
  - Create a configuration xml file

    <Context path="/midp" **docBase**="midp" **reloadable**="true"**/>**
  - Deploy the web application using the above XML file

Deploying the Servlet

- Create **webapps/midp/WEB-INF/web.xml**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
    Application 2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <servlet>
    <servlet-name>bob</servlet-name>
    <servlet-class>HitServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>bob</servlet-name>
    <url-pattern>/hits</url-pattern>
  </servlet-mapping>
</web-app>
```

Tells Tomcat to map the servlet HitServlet to the path /hits

Testing the Servlet

- Connect to http://localhost:8080/midp/hits
- You should see the output of HitServlet, i.e., a count that is increased when you reload the page
- Hits: 9
- Hits: 10
- Hits: 11
- …

Hooking up a MIDlet to the Servlet

- Hooking up
  - A MIDlet can connect to the world via HTTP
  - A Servlet is available to the world via HTTP
  - So, it is easy to connect a MIDlet to a Servlet
- Start KToolbar, open the MIDlet project (from Part I), i.e, **`HelloSuite`**
- Create a new MIDlet that connects to the servlet, retrieves its output, and displays it.
- Source codes of HitMIDlet (next 2 slides)

```
import java.io.*;
import javax.microedition.io.*;
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class HitMIDlet extends MIDlet implements CommandListener {
    private Display mDisplay;
    private Form mMainForm;
    private StringItem mMessageItem;
    private Command mExitCommand, mConnectCommand;
    public HitMIDlet() {
        mMainForm = new Form("HitMIDlet");
        mMessageItem = new StringItem(null, "");
        mExitCommand = new Command("Exit", Command.EXIT, 0);
        mConnectCommand = new Command("Connect", Command.SCREEN, 0);
        mMainForm.append(mMessageItem);
        mMainForm.addCommand(mExitCommand);
        mMainForm.addCommand(mConnectCommand);
        mMainForm.setCommandListener(this); }
    public void startApp() { mDisplay = Display.getDisplay(this); mDisplay.setCurrent(mMainForm); }
    public void pauseApp() {}
    public void destroyApp(boolean unconditional) {}
    public void commandAction(Command c, Displayable s) {
        if (c == mExitCommand) notifyDestroyed();
        else if (c == mConnectCommand) {
            Form waitForm = new Form("Waiting...");
            mDisplay.setCurrent(waitForm);
            Thread t = new Thread() {
                public void run() { connect(); }
            };
        t.start();
    }
}
```

```
private void connect() {
    HttpConnection hc = null;
    InputStream in = null;
    String url = getAppProperty("HitMIDlet.URL");
    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();
            // Show the response to the user.
            String s = new String(raw, 0, length);
            mMessageItem.setText(s);
    }
    catch (IOException ioe) { mMessageItem.setText(ioe.toString()); }
    mDisplay.setCurrent(mMainForm);
    }
}
```

To make HitMIDlet start working

- Click on Settings in KToolbar
- Select the MIDlets tab
- Click on Add and fill in “HitMIDlet” for both MIDlet name and class name
  - You can leave ICON blank
- Click OK
- Both HelloMIDlet and HitMIDlet must be listed

To make HitMIDlet start working

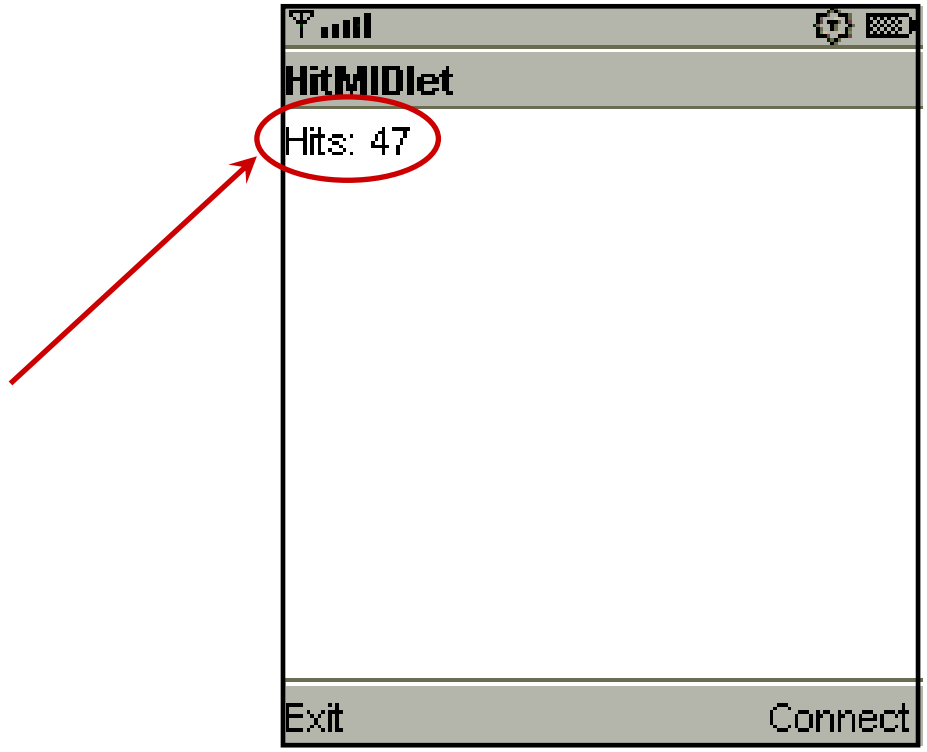
- Need to define a system property that HitMIDlet uses as URL for network connection
- Click on Settings | User Defined | Add
- Fill in “HitMIDlet.URL” as the property name
  - Value: http://localhost:8080/midp/hits
- Click OK

To make HitMIDlet start working

- In Java Wireless Toolkit
  - Click Build
  - Click Run
  - Select Connect
- HitMIDlet invokes HitServlet
- The hit counter value is displayed in the mobile phone emulator

Success!!!

- The result of HitServlet is display by HitMIDlet in the device emulator



Conclusion

- This tutorial shows how to develop an end-to-end Java wireless application
  - Part I
    - setup and use a MIDlet development environment
    - create a MIDlet and run and test it
  - Part II
    - configure and use a servlet development environment
    - Write and deploy a servlet
    - Hook up MIDlet with Servlet (end-to-end application)