

Develop mobile apps with Personal Information Management

Configure the Eclipse SDK to develop a J9/JSR-75 PIM application

Skill Level: Intermediate

[Anderson Fraga \(anderson_fraga@ca.ibm.com\)](mailto:anderson_fraga@ca.ibm.com)
Software Engineer
IBM

18 Jul 2006

This tutorial takes you through the steps required to successfully build a Personal Information Management (PIM) Mobile Information Device Profile 2.0 (MIDP 2.0) application using well-known products, such as Eclipse, the J9 Java™ Virtual Machine, and the JSR-75 PIM implementation.

Section 1. Before you start

About this tutorial

The Java Specification Request (JSR) 75 includes the Personal Information Management (PIM) package, along with the FileConnection (FC) package. The PIM package lets developers access a user's contacts, to-dos, and calendar events available on cell phones, PDAs, and any other Java-capable portable device.

After reading this tutorial, you will be able to use Eclipse to develop Java 2 Platform, Micro Edition (J2ME) applications that let users access and interact with the personal information available on their devices. This is an intermediate-level tutorial aimed at developers with previous experience in Java, Eclipse, and embedded

development, but does not require any JSR-75 experience.

Prerequisites

- **Eclipse SDK:** You can download Eclipse from the [Eclipse Foundation](#) Web site (latest version at time of writing is 3.1.2).
 - **J9 MIDP 2.0 Runtime for Windows Mobile 5.0:** Download evaluation versions from [WebSphere Everyplace Micro Environment Trial and Evaluation Runtimes](#) (latest version at time of writing is 6.1).
 - **J9 MIDP 2.0 libraries for Windows:** Find instructions on how to download and install in the [Installation and Configuration](#) section.
 - **J9 PIM Optional Package Libraries:** Find instructions on how to download and install in the [Installation and Configuration](#) section.
 - **J9 Launcher plug-in for Eclipse:** Find instructions on how to download and install in the [Installation and Configuration](#) section.
-

Section 2. Installation and configuration

Install the Eclipse SDK

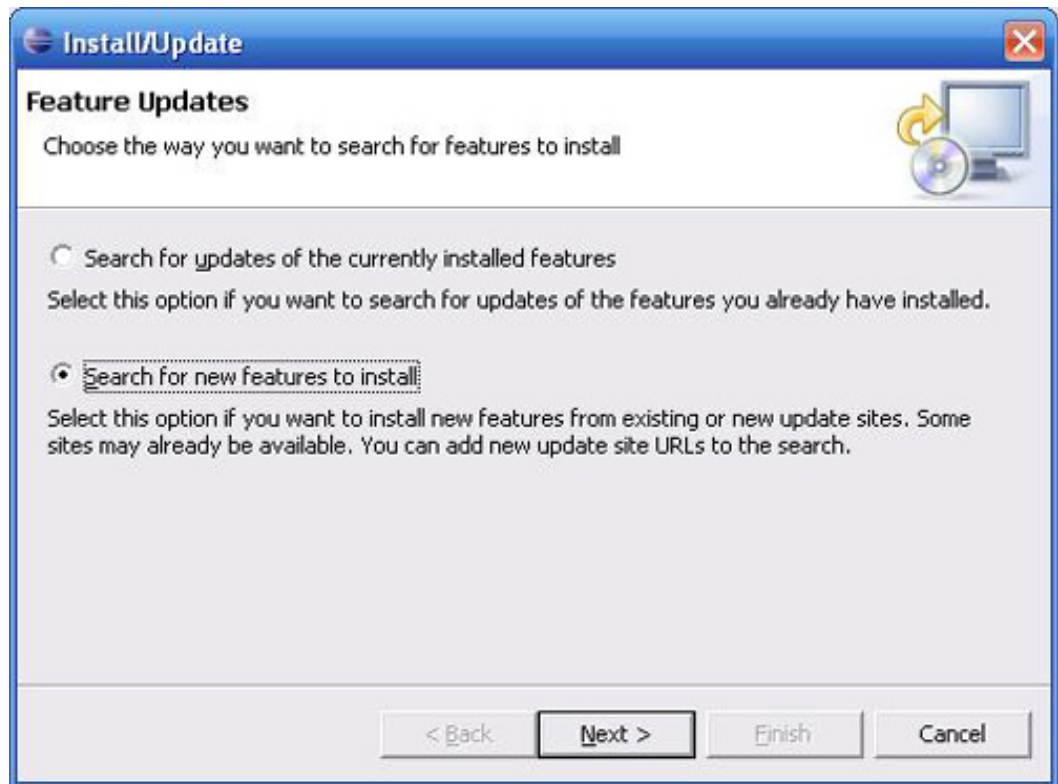
I use Eclipse as the IDE to develop the PIM application. After you have downloaded the Eclipse SDK, extract the zip file to the location where you want to install the application. You don't need any further installation. You may want to create a link to eclipse.exe on your desktop after you unzip the SDK.

Install MIDP 2.0 and PIM Optional Package

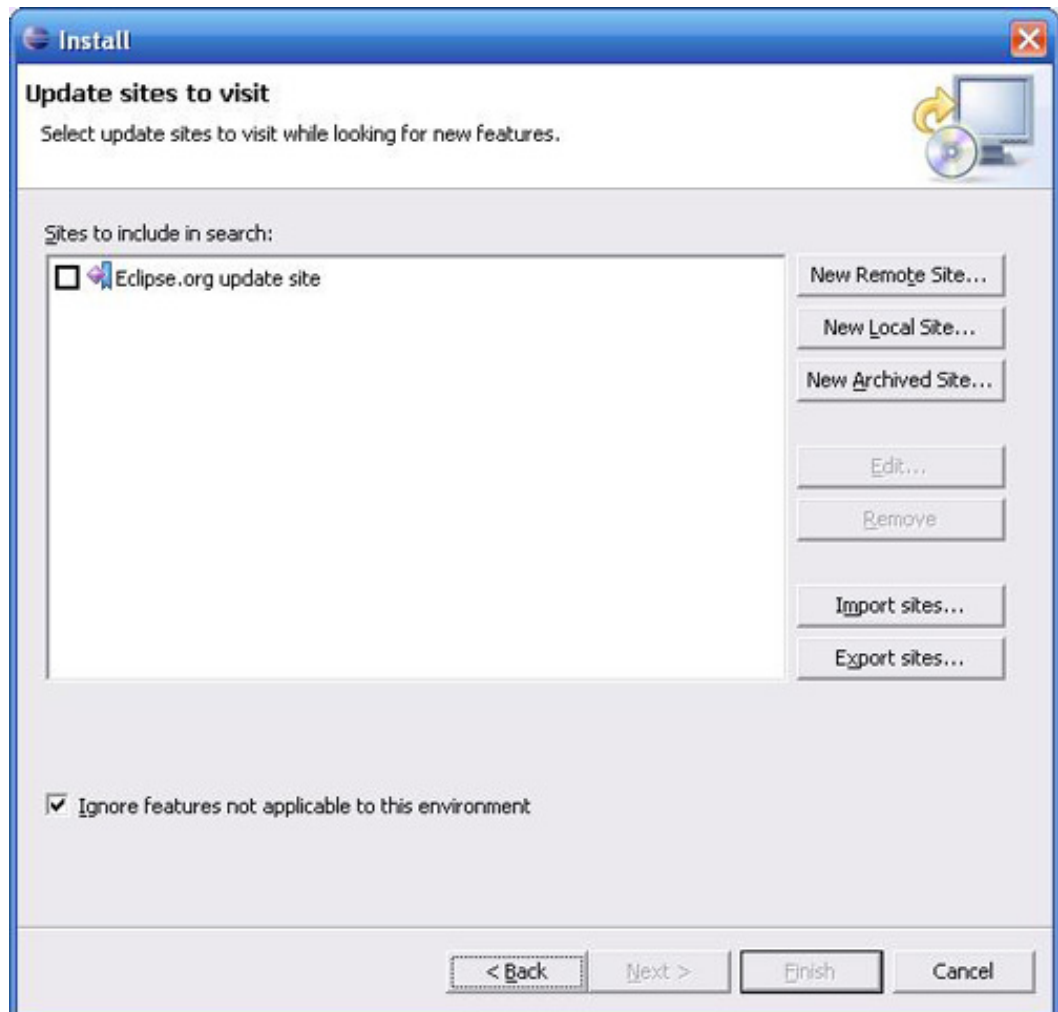
The IBM J9 Java Virtual Machine and class libraries are available through WebSphere® Everyplace Micro Environment. To download WebSphere Everyplace Micro Environment, open Eclipse and follow these instructions:

1. On the Eclipse main menu, navigate to **Help > Software Updates > Find and Install**. On the Install/Update window, select **Search for new features to install** and click **Next** (see [Figure 1](#)).

Figure 1. Eclipse's Install/Update window



2. On the Install/Update window, you'll need to add two new remote sites by clicking **New Remote Site** (see [Figure 2](#)).
Figure 2. Eclipse's New Remote Site window

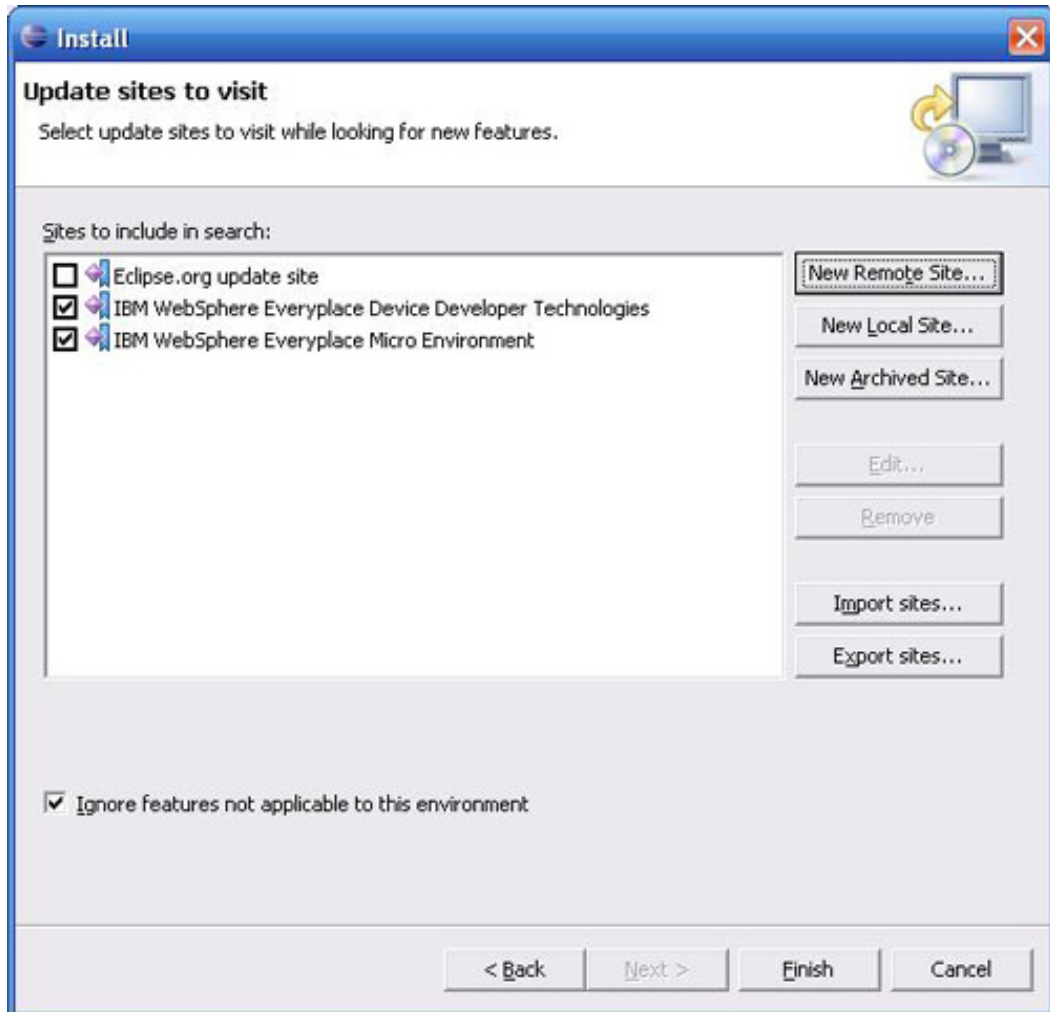


Here are the names and URLs of the new remote sites you'll need to add:

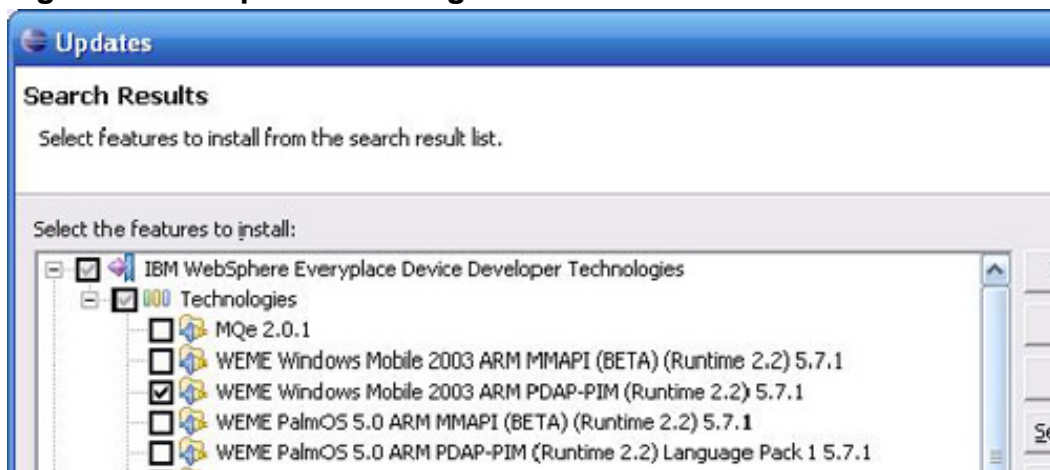
- Name: IBM WebSphere Everyplace Micro Environment
URL: <http://www.ibm.com/software/pervasive/wsdd/updates/571/weme>
- Name: IBM WebSphere Everyplace Device Developer Technologies
URL: <http://www.ibm.com/software/pervasive/wsdd/updates/571/techs>

3. Add a check mark to the new remote sites you just added and click **Finish**.

Figure 3. Select new remote sites

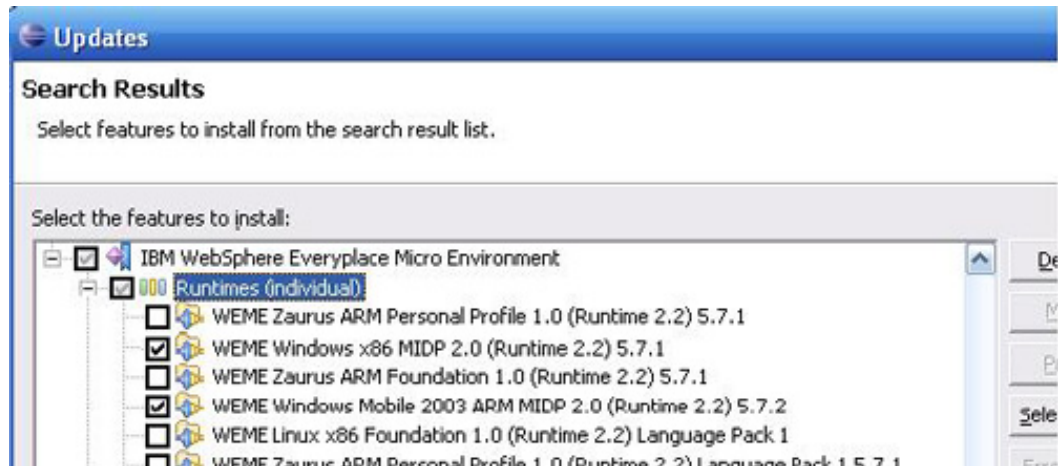


4. Expand **IBM WebSphere Everyplace Micro Environment**, expand **Technologies**, and add a check mark to **WEME Windows Mobile 2003 ARM PDAP-PIM (Runtime 2.2)**, as [Figure 4](#) shows.
Figure 4. PIM Optional Package install



- Also, expand **IBM WebSphere Everyplace Micro Environment**, expand **Runtimes (Individual)**, and add a check mark in **WEME Windows x86 MIDP 2.0 (Runtime 2.2) 5.7.1** as well as another check mark in **WEME Windows Mobile 2003 ARM MIDP 2.0 (Runtime 2.2) 5.7.2** (see [Figure 5](#)).

Figure 5. WebSphere Everyplace Micro Environment installation



Click **Next** and perform the installations.

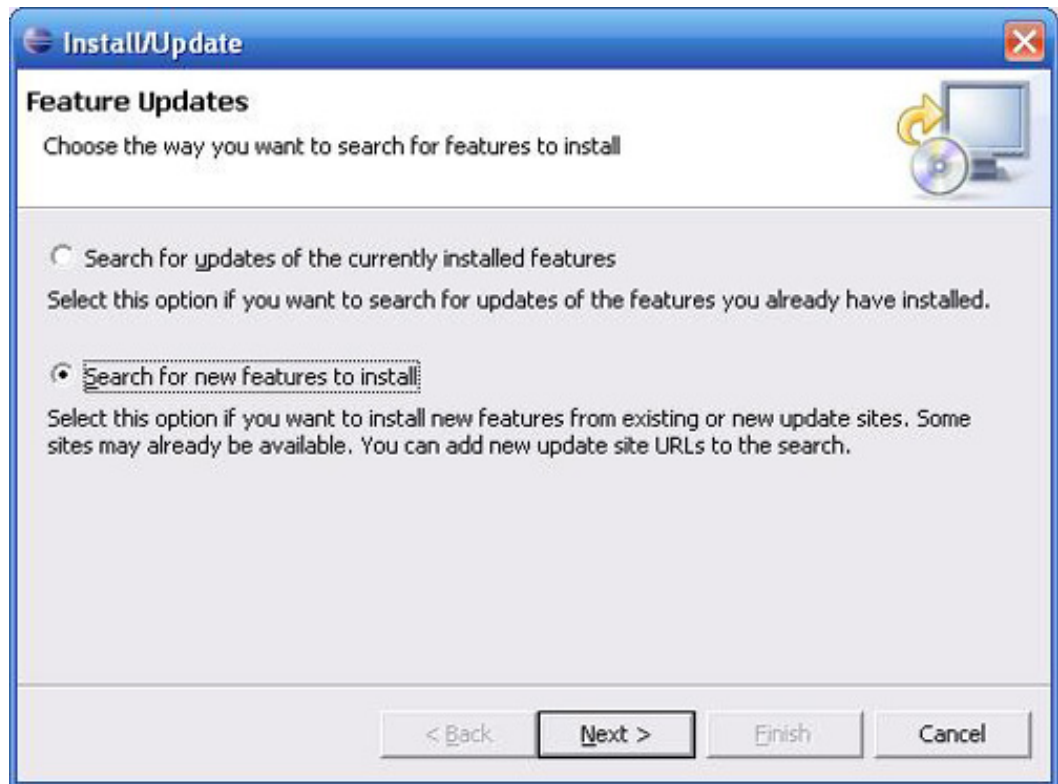
You may be asked to restart the workbench after the installations are complete; select **Yes**.

Install J9 Launcher plug-in

The J9 Launcher plug-in lets Eclipse recognize J9's VM as a valid Java Runtime Environment (JRE); it is available through the IBM WebSphere Everyplace Device Developer Tooling. To install the J9 Launcher plug-in:

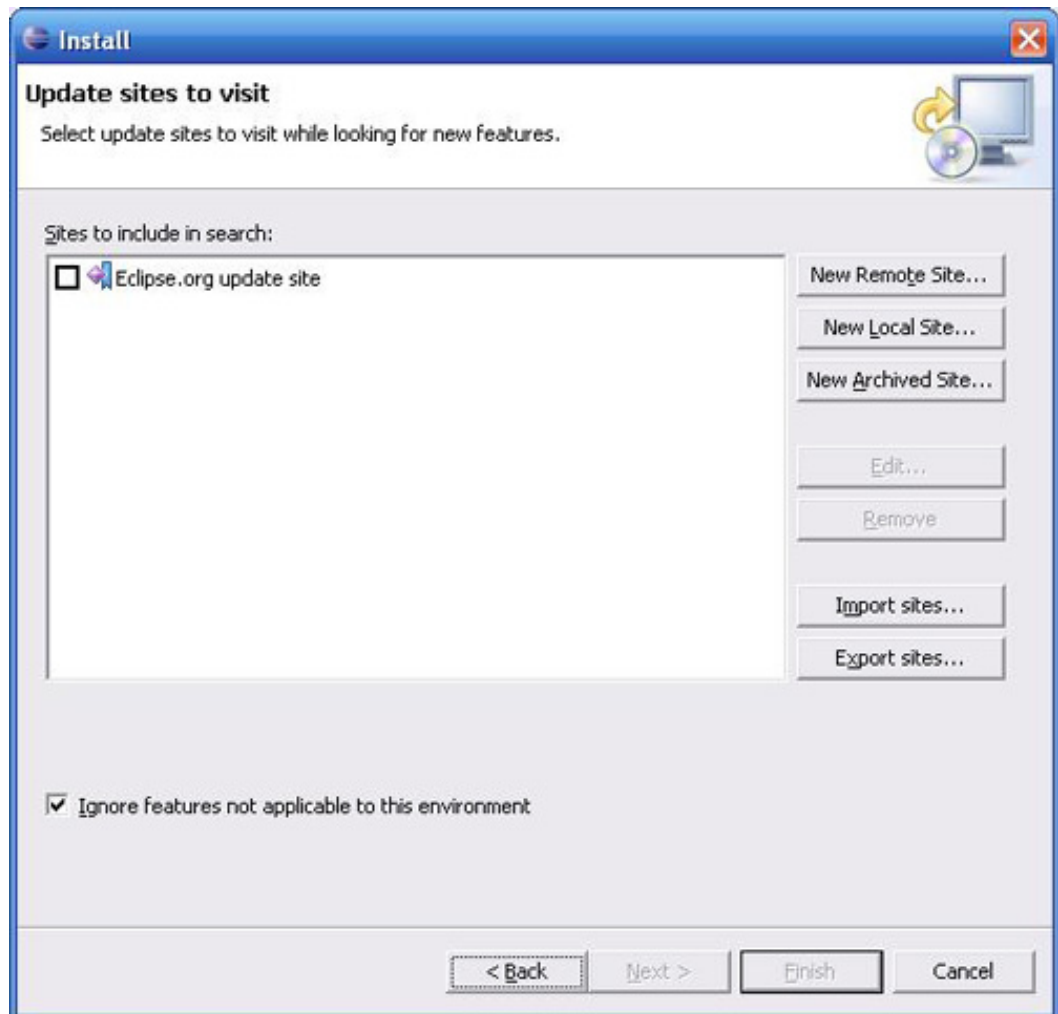
- On the Eclipse main menu, go to **Help > Software Updates > Find and Install**. On the Install/Update window, select **Search for new features to install** and click **Next** (see [Figure 6](#)).

Figure 6. Eclipse's Install/Update window



2. On the Install/Update window, you will need to add a new remote site by clicking **New Remote Site**.

Figure 7. Eclipse's New Remote Site window

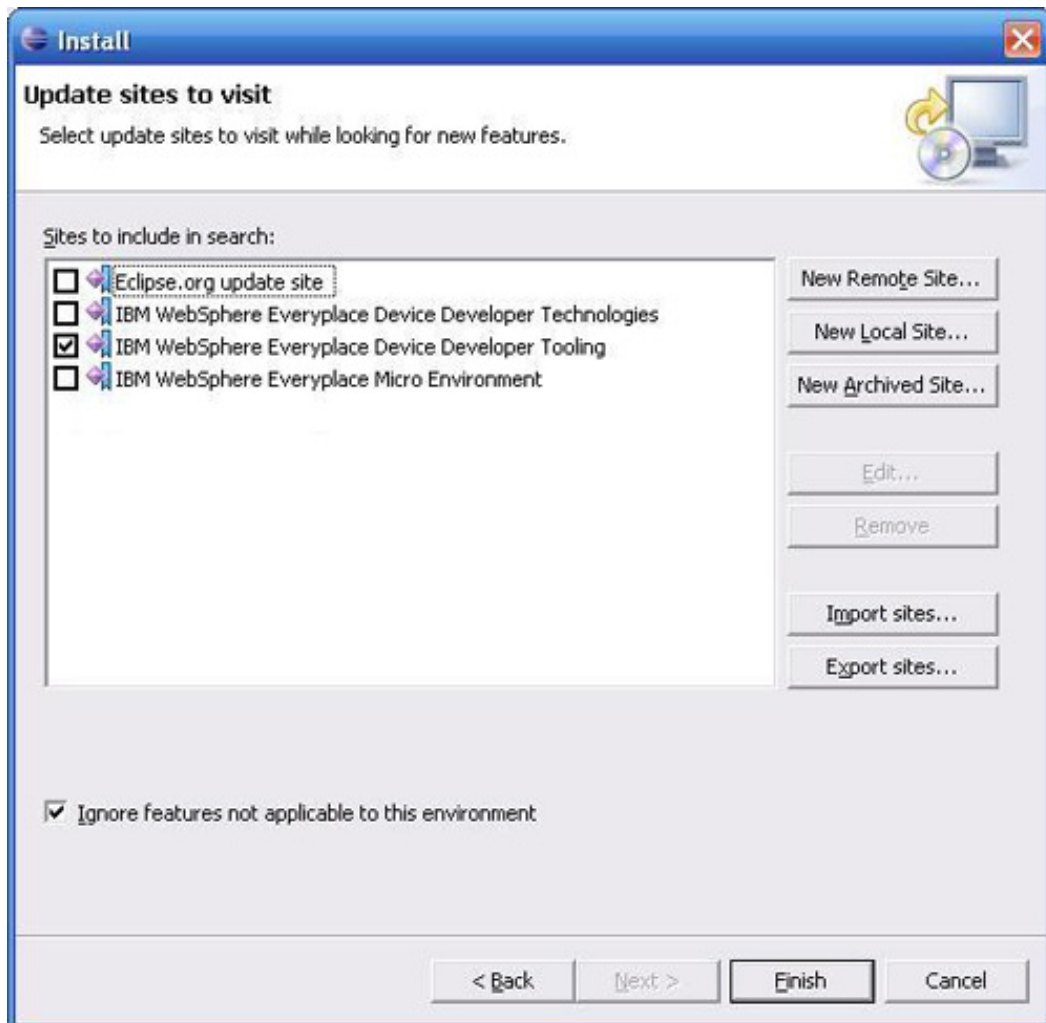


The name and URL of the new remote site you'll need to add is:

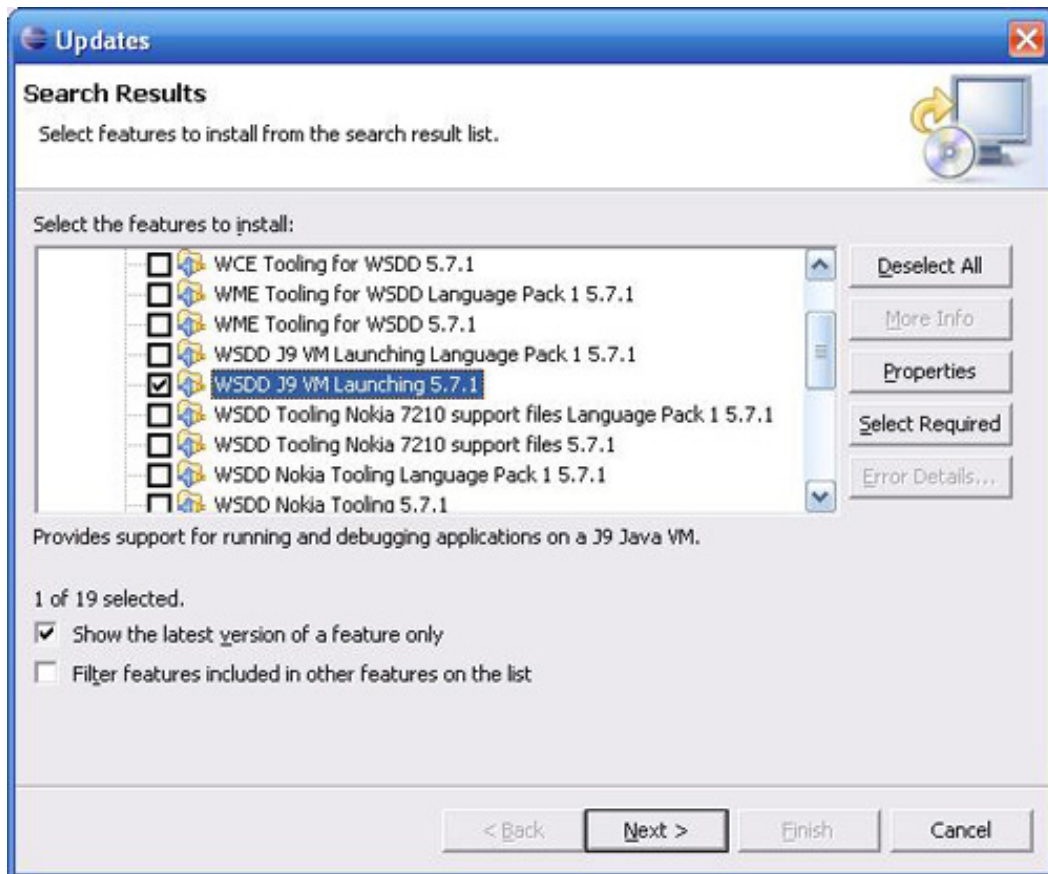
- Name: IBM WebSphere Everyplace Device Developer Tooling
URL:
<http://www.ibm.com/software/pervasive/wsdd/updates/571/tooling>

3. Add a check mark to the new remote sites you just added and click **Finish**.

Figure 8. Select new remote site



4. Expand **IBM WebSphere Everyplace Device Developer Tooling**, expand **Tools**, and add a check mark to **WSDD J9 VM Launching 5.7.1**.
Figure 9. J9 Launcher plug-in install



Click **Next** and perform the installation.

You may be asked to restart the workbench after the installation process is complete; select Yes.

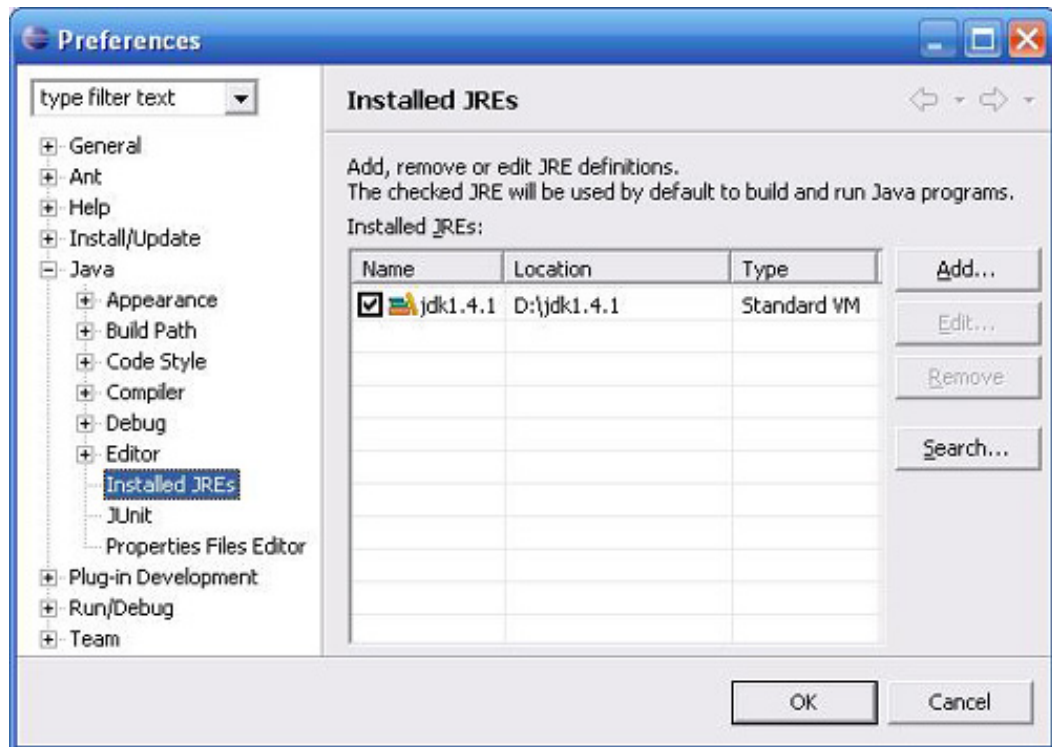
Configure Eclipse to use J9

For Eclipse to use J9, you must first install the J9 Launcher plug-in in the previous section.

To configure Eclipse to use J9:

1. Select **Window > Preferences**. The Preferences window opens. Expand the tree on the left panel and navigate to **Java > Installed JREs**, as [Figure 10](#) shows.

Figure 10. Eclipse Preferences window



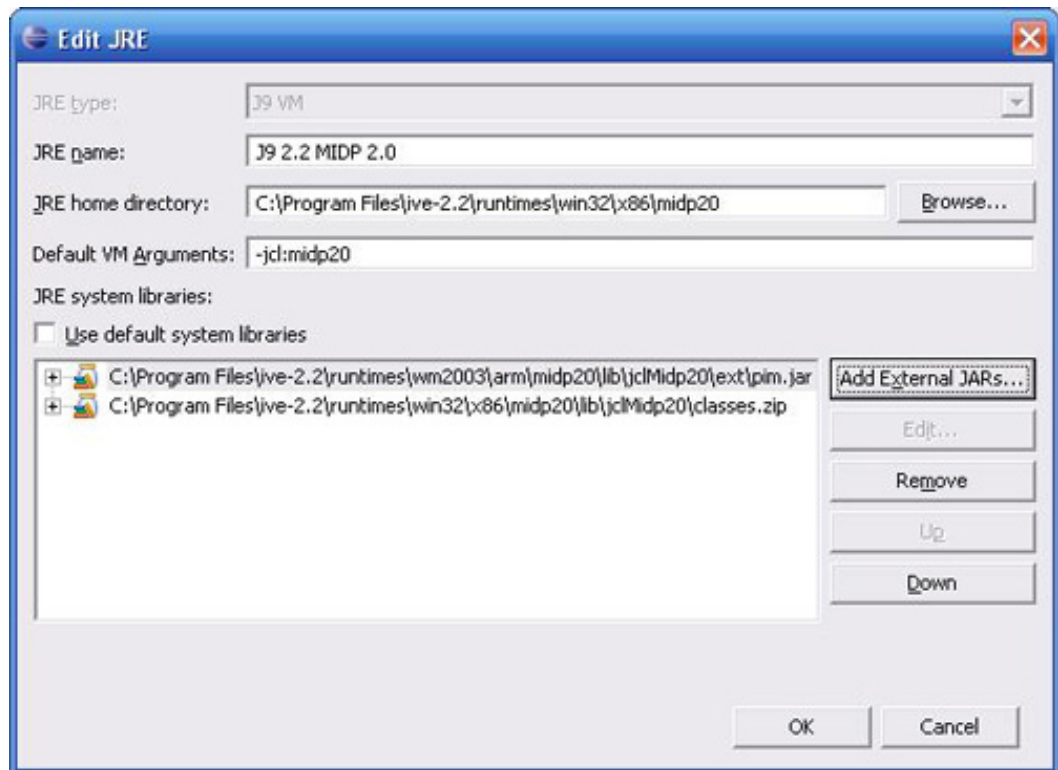
2. Click **Add** to open the **Add JRE** window. On this window, set the options as follows:

- JRE Type: Select J9 VM
- JRE Name: Type "J9 MIDP 2.0" or any name you'd like to use to identify this JRE
- JRE home directory: Browse to the directory where Eclipse installed WebSphere Everyplace Micro Environment (it should be in "C:\Program Files\ive2.2\runtimes\win32\x86\midp20").
- Default VM Arguments: type "-jcl:midp20"

Also, uncheck the option **Use default system libraries**, click **Add External JARs**, and add the following external classes:

- WebSphere Everyplace Micro Environment MIDP 2.0 classes.zip: You can find this in "C:\Program Files\ive-2.2\runtimes\win32\x86\midp20\classes.zip".
- PDAP Optional Package jar file: You can find this in "C:\Program Files\ive-2.2\runtimes\wm2003\arm\midp20\lib\jclMidp20\ext\pim.jar".

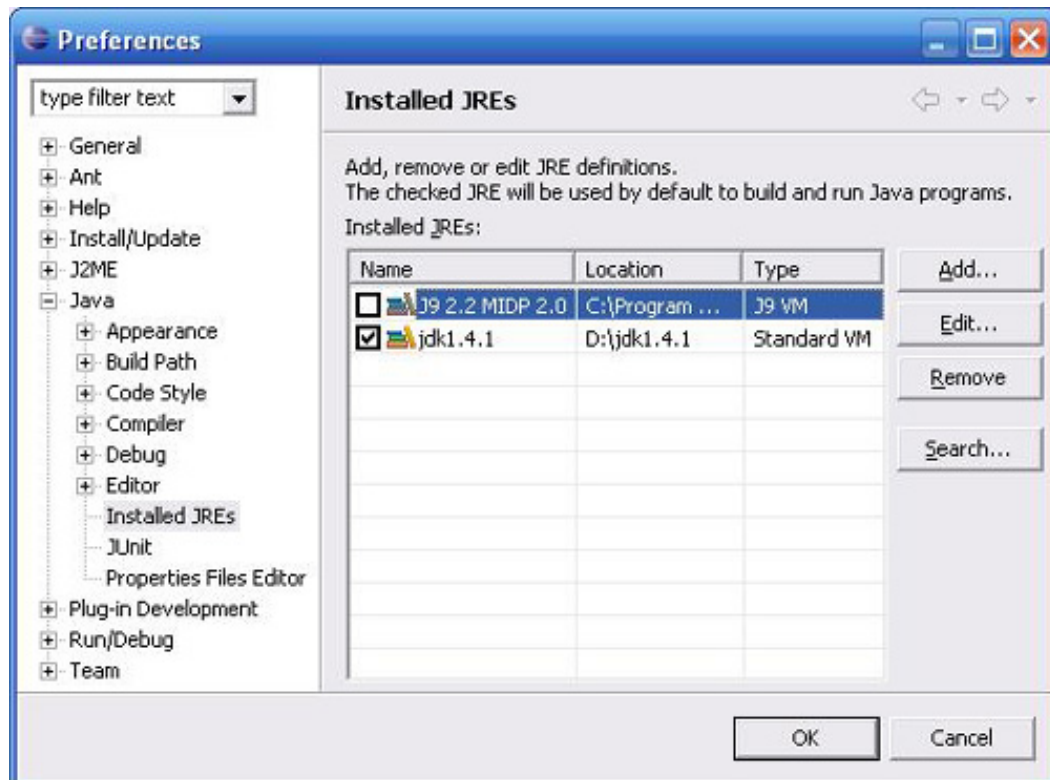
Figure 11. Edit JRE window



You should easily find the directories above, granted you successfully followed the instructions in the [Install MIDP 2.0 and PIM optional package](#) section. However, if you do not find the path, try searching your file system for `pim.jar` and it will point you to the correct path.

3. Click **OK**. You will see that you added the **J9 2.2 MIDP 2.0** and can use it on your MIDP 2.0 PIM application.

Figure 12. JRE window



Click **OK** again to go back to the Eclipse workbench.

Section 3. The PIM Optional Package API

The PIM APIs are included as an optional package to the J2ME libraries. Optional packages let you use additional functions not previously supported by a given J2ME configuration/profile.

PIM optional package discovery

To use the PIM API, you must first make sure that the PIM Optional Package is installed on the target platform.. You can use the system property `microedition.pim.version` to determine this, as [Listing 1](#) shows.

Listing 1. PIM Optional Package Discovery

```
if (System.getProperty("microedition.pim.version") == null) {
    /* PIM Optional Package is NOT installed */
}
```

```
} else {  
    /* PIM Optional Package is installed */  
}
```

PIM lists

PIM lists to hold zero or more PIM items. You can make these lists of the following types:

- Contact list: Holds zero or more Contact items
- To-Do list: Holds zero or more To-Do items
- Event list: Holds zero or more Calendar Event items

The PIM class is the main point of access to the various PIM lists. You can use it to access all the supported PIM lists.

[Listing 2](#) shows how to open a Contact list.

Listing 2. Opening a PIM list

```
/* Open a Contact List */  
ContactList contactList;  
try {  
    contactList = (ContactList)  
PIM.getInstance().openPIMList(PIM.CONTACT_LIST,  
                               PIM.READ_WRITE);  
} catch (PIMException pex) {  
    /* error */  
}
```

On the `openPIMList()` call, first specify the type of list you want to open and then the access level required. A system usually reports this access level to the user at run time, asking for the proper permission; in this case, the user may grant or deny the access.

PIM items

PIM items are contacts, to-dos, and calendar events that your device displays. After you open a PIM list, you can create and add PIM items to the list. [Listing 3](#) shows how you would create a contact item and add it to your contact list:

Listing 3. Create a contact item

```
/* Create a Contact */  
Contact contactItem = contactList.createContact();
```

```

/* add some fields (explained in next section) */
...
try {
    /* Commit the Contact Item */
    contactItem.commit();
} catch (PIMException pex) {
    /* error committing contact item */
}

try {
    /* Close Contact List */
    contactList.Close();
} catch (PIMException pex) {
    /* error closing contact list */
}

```

In [Listing 3](#), you first create a Contact Item object using an open contact list. Next, you can add the data fields that describe the contact and then call `commit()` to save the contact item to your device's contact list. At last, you close the contact list. That's the pattern you will see when adding a new PIM item to any of the PIM list types. However, you must change your list and item types accordingly, as you cannot add a contact item to an event list.

PIM fields

Supported fields for a given PIM item vary depending on the implementation and platform. Because of this, you must explicitly check all fields for support prior to using them. The PIM API provides a way to check for field support through the `isSupportedField()/isSupportedArrayElement()` methods, accessible through a PIM List object.

Listing 4. Adding fields to a contact item

```

/* Create a Contact */
Contact contactItem = contactList.createContact();

/* add some fields */
String[] nameArr = new
String[contactList.stringArraySize(Contact.NAME)];

/* add name */
if (contactList.isSupportedArrayElement(Contact.NAME,
Contact.NAME_FAMILY))
    nameArr [Contact.NAME_FAMILY] = "Fraga";
if (contacts.isSupportedArrayElement(Contact.NAME,
Contact.NAME_GIVEN))
    nameArr [Contact.NAME_GIVEN] = "Anderson";
contactItem.addStringArray(Contact.NAME, PIMItem.ATTR_NONE,
nameArr);

/* add telephone */
if (contactList.isSupportedField(Contact.TEL))
    contactItem.addString(Contact.TEL, Contact.ATTR_NONE,
"613-123-4567");

```

```

try {
    /* Commit the Contact Item */
    contactItem.commit();
} catch (PIMException pex) {
    /* error committing contact item */
}

try {
    /* Close Contact List */
    contactList.Close();
} catch (PIMException pex) {
    /* error closing contact list */
}
    
```

All fields have a data type associated with them. [Tables 1](#) through [3](#) show the standard fields and data types for the PIM items.

Table 1. CONTACT standard fields

Contact Fields	Data Type
NAME, ADDR	PIMItem.STRING_ARRAY
EMAIL, FORMATTED_NAME, NICKNAME, NOTE, ORG, TEL, TITLE, UID, URL, PHOTO_URL, PUBLIC_KEY_STRING	PIMItem.STRING
BIRTHDAY, REVISION	PIMItem.DATE
PHOTO, PUBLIC_KEY	PIMItem.BINARY
CLASS	PIMItem.INT

Table 2. EVENT standard fields

Event Fields	Data Type
LOCATION, NOTE, SUMMARY, UID	PIMItem.STRING
END, REVISION, START	PIMItem.DATE
ALARM, CLASS	PIMItem.INT

Table 3. TO-DO standard fields

ToDo Fields	Data Type
NOTE, SUMMARY, UID	PIMItem.STRING
CLASS, PRIORITY	PIMItem.INT
COMPLETION_DATE, DUE, REVISION	PIMItem.DATE
COMPLETED	PIMItem.BOOLEAN

PIM attributes

Attributes help describe information stored in a field. You can set them when you create a field or if you edit an already existing field. For example, you may specify the following attributes when creating a field of type `Contact.TEL`:

- `ATTR_FAX`: Associates the TEL field to a fax number
- `ATTR_HOME`: Associates the TEL field to a home phone number
- `ATTR_MOBILE`: Associates the TEL field to a mobile phone number

You may also specify multiple attributes with the `|` (OR) bitwise operator.

[Listing 5](#) shows how to create a home phone field and a work mobile phone field for a given contact.

Listing 5. Adding an attribute to a field

```
/* add contact's home phone field */
if (contactList.isSupportedField(Contact.TEL)
&& contactList.isSupportedAttribute(Contact.TEL,
Contact.ATTR_HOME))
    aContact.addString(Contact.TEL, Contact.ATTR_HOME,
"613-123-4567");

/* add contact's work phone field */
if (contactList.isSupportedField(Contact.TEL)
&& contactList.isSupportedAttribute(Contact.TEL,
Contact.ATTR_WORK))
    aContact.addString
(Contact.TEL, Contact.ATTR_WORK|Contact.ATTR_MOBILE,
"613-765-4321");
```

You can use the attribute `ATTR_NONE` to specify that the field has no attributes.

PIM extended fields and attributes

Extended fields and attributes provide platform-specific field support. Those are fields the API does not define, but that the platform's native implementation supports.

An example of a possible extended field would be `Contact.RING_TONE`. This field is not one of the contact fields that the API lists, but the Windows Mobile 5.0 native implementation does support it, so it is a good candidate for an extended field. I can explain extended attributes in the exact same way, but instead of a field it pertains to attributes.

The JSR-75 PIM implementation (in this case IBM's J9) defines extended fields or attributes. However, you are responsible for discovering and using them wherever possible. You can use the `getSupportedFields()` and `getSupportedAttributes()` APIs to get a list of all supported fields and attributes respectively, including the extended ones.

Section 4. Develop a PIM MIDlet application

This section guides you through the steps required to build and deploy a PIM mobile application to a Windows mobile device running J9.

The BirthdayReminder MIDlet

The BirthdayReminder MIDlet is a small application written using MIDP 2.0 and the PIM Optional Package. It lets the user enter friends' birthdays as calendar events and notifies him or her when a friend's birthday approaches. To keep this tutorial on its main topic, I won't get into the details of how to write a MIDlet. However, there are many good tutorials that cover this topic; check out the rest of the Wireless section on developerWorks to learn more.

Creating the BirthdayReminder project

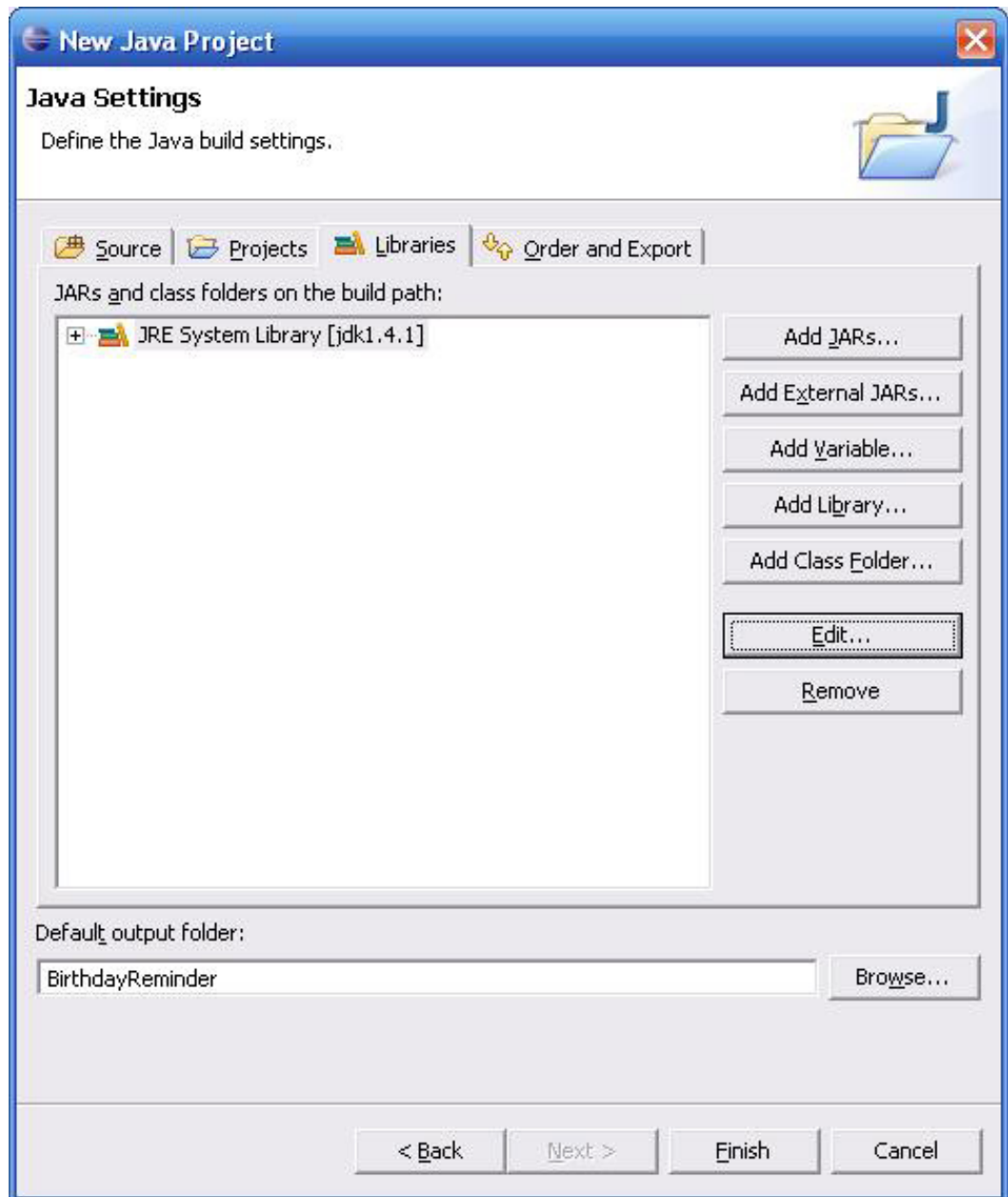
To create the BirthdayReminder Java project on Eclipse:

1. Go to **File > New > Project...** and select **Java Project** on the wizard window.
2. Enter `BirthdayReminder` as the project's name and click **Next**.

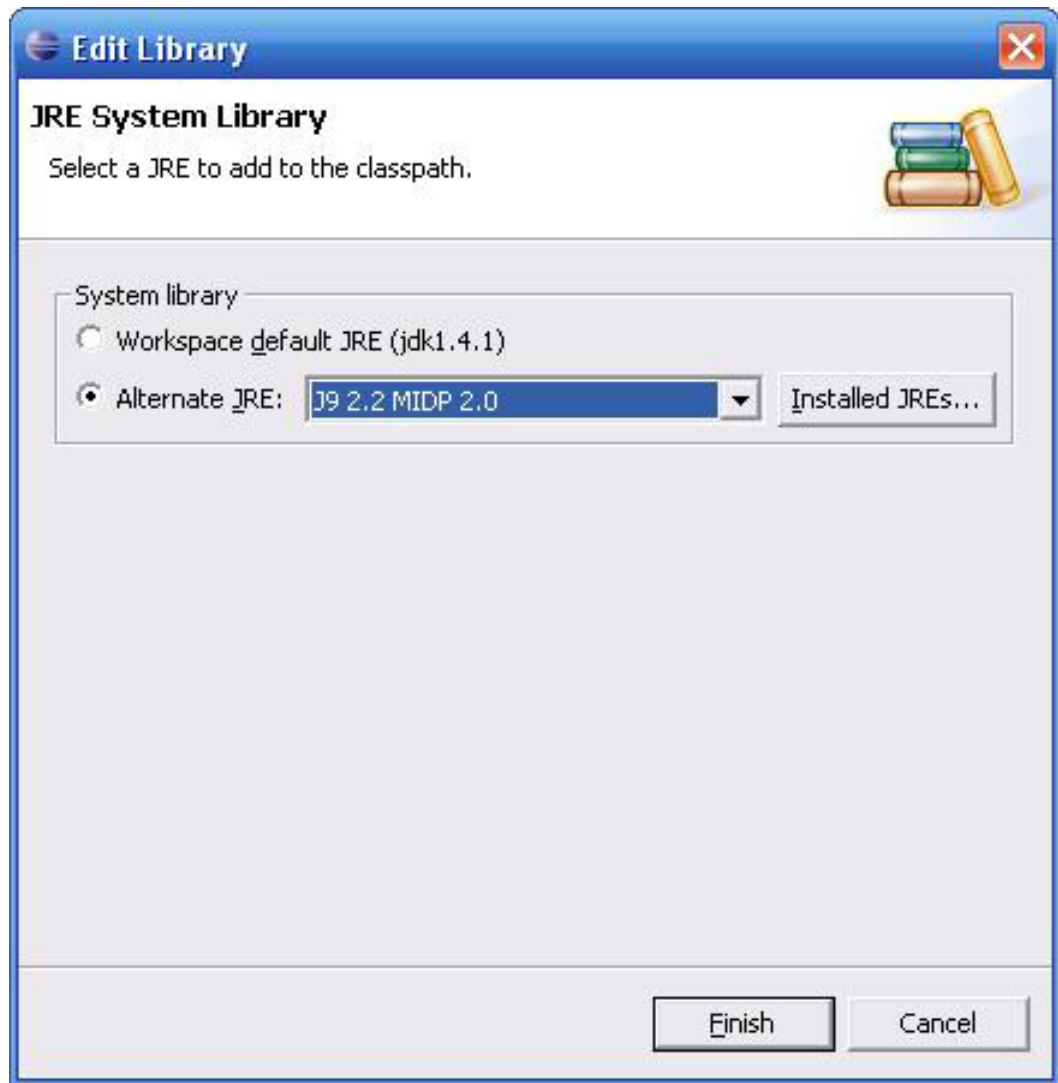
Figure 13. New project



3. On the Libraries tab, select the available library and click **Edit**.
- Figure 14. New project library**



4. On the Edit Library window, select **Alternate JRE** and select the J9 2.2 MIDP 2.0 library, as [Figure 15](#) shows.
Figure 15. Edit Library window

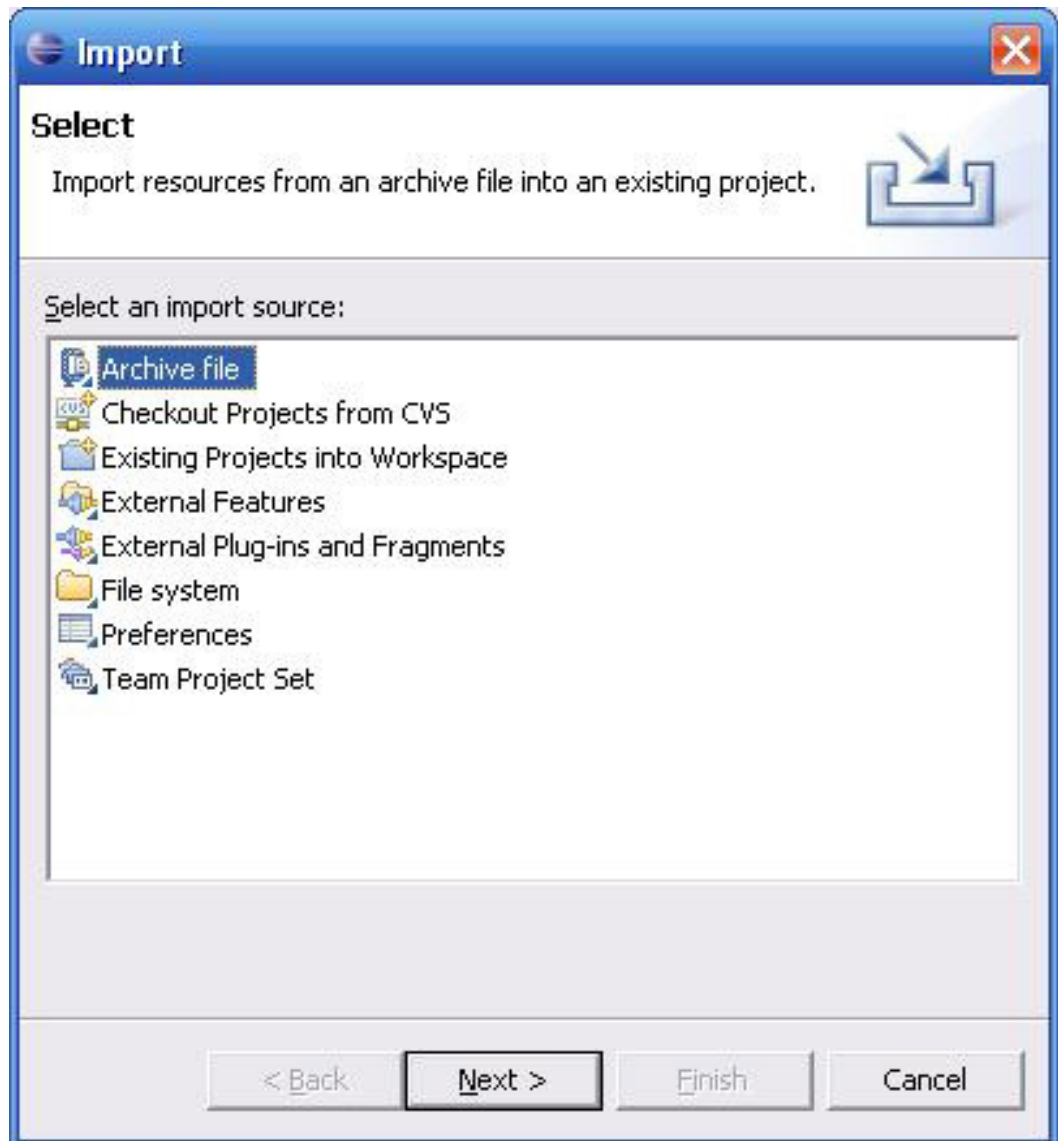


5. Click **Finish**.

Import the BirthdayReminder source code

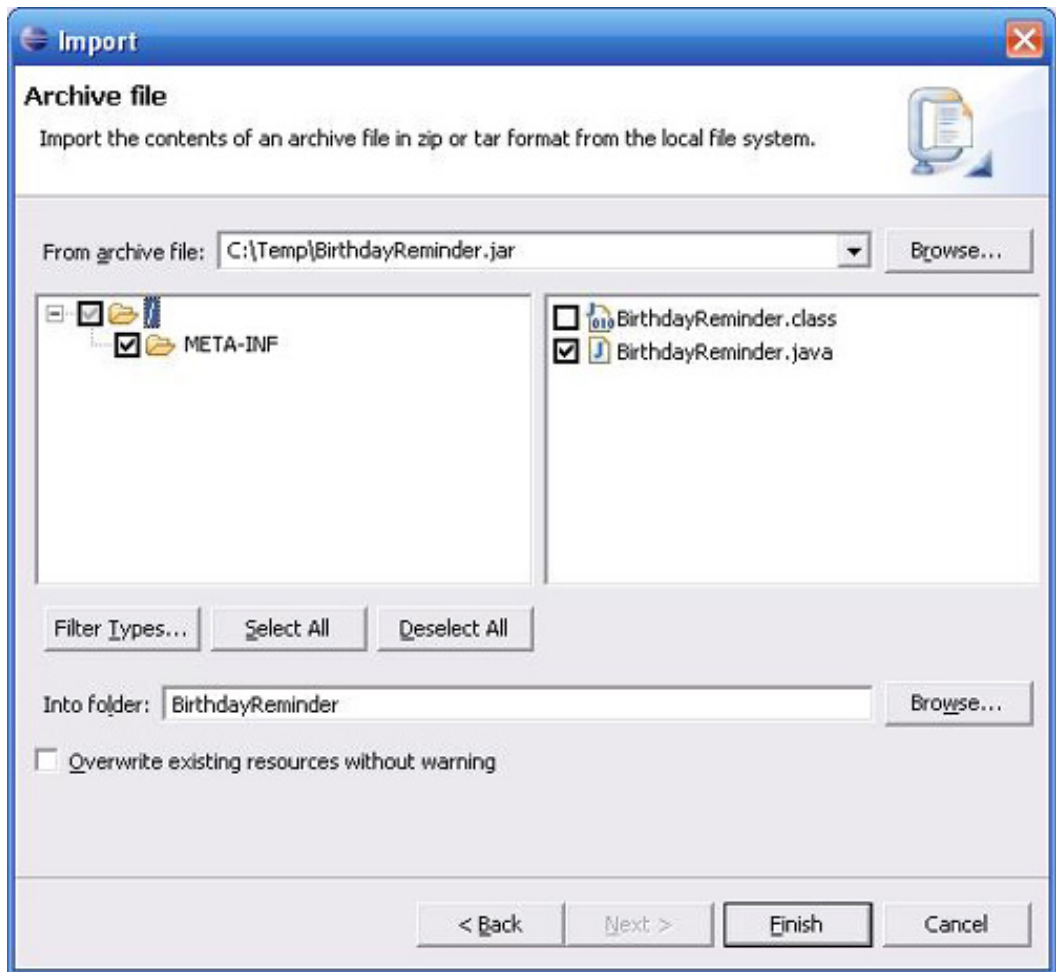
Now that you have created the BirthdayReminder project, import the source code so you can use it as you read the tutorial.

1. Inside Eclipse, right-click the BirthdayReminder project and select **Import...** to open the import wizard. Select **Archive file** and click **Next**.
Figure 16. Importing source code



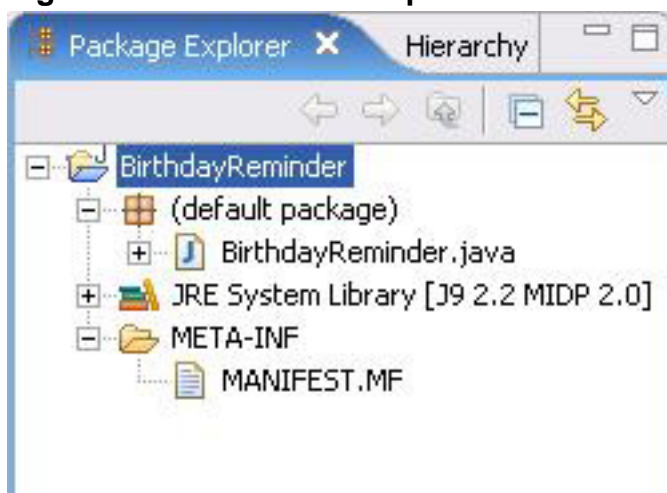
2. Next, extract the BirthdayReminder.zip file to a temp folder. Browse to the BirthdayReminder.jar file just extracted and click **Finish**, as [Figure 17](#) shows.

Figure 17. Importing from the archive



3. After you import the source code for BirthdayReminder, you should see it on the Eclipse's Package Explorer, as [Figure 18](#) shows.

Figure 18. Source code imported



An overview of the BirthdayReminder source code

Browsing the source code, you see that the MIDlet first checks if you installed the PIM Optional Package (see [Listing 6](#)).

Listing 6. Check for PIM installation

```
public boolean hasPIMLibrary() {
    if (System.getProperty("microedition.pim.version") == null)
        return false;
    else
        return true;
}
```

If the Optional Package is not installed, the user is notified.

Listing 7. Notify user if PIM library is not installed

```
protected void startApp() throws MIDletStateChangeException {
    display = Display.getDisplay(this);
    cmdExit = new Command("Exit", Command.EXIT, 0);
    cmdOk = new Command("OK", Command.OK, 0);

    // check if PIM Library is installed
    if (!hasPIMLibrary())
        display.setCurrent(getPIMNotInstalledForm());
    else
        display.setCurrent(getForm());
}
```

However, if the system finds the Optional Package, it presents the user with the BirthdayReminder form. The code relevant to the PIM API is contained within the `addBirthdayReminder()` method. This method first opens an `EventList` and creates a new `Event` object for the birthday that the user enters.

Listing 8. Create a birthday event

```
// open the default Event List
eventList = (EventList)
PIM.getInstance().openPIMList(PIM.EVENT_LIST, PIM.READ_WRITE);

// create an Event
event = eventList.createEvent();

// set a few Event properties
if (eventList.isSupportedField(Event.SUMMARY))
    event.addString(Event.SUMMARY, Event.ATTR_NONE, firstName +
" " +
        lastName + " birthday");
if (eventList.isSupportedField(Event.NOTE))
    event.addString(Event.NOTE, Event.ATTR_NONE, "Don't forget
" +
        firstName + " " + lastName + " birthday");
```

```
if (eventList.isSupportedField(Event.START))
    event.addDate(Event.START, Event.ATTR_NONE,
birthday.getTime());
if (eventList.isSupportedField(Event.END))
    event.addDate(Event.END, Event.ATTR_NONE,
birthday.getTime());
```

The method then sets a few fields that describe the Event, specifically `Event.SUMMARY` and `Event.NOTE`. It also sets an `Event.START` and `Event.END` with the exact same date. This will make the Event an all-day event.

With the required fields set, the method then creates a yearly `RepeatRule` for the birthday, as [Listing 9](#) shows.

Listing 9. Setting the event's RepeatRule

```
// create a repeat rule object and set properties to repeat the
same Event
// on the same day and month of the year for an unlimited
number of years
rule = new RepeatRule();
rule.setInt(RepeatRule.FREQUENCY, RepeatRule.YEARLY);
rule.setInt(RepeatRule.MONTH_IN_YEAR,
mapCalendarRepeatMonth(cal.get(Calendar.MONTH)));
rule.setInt(RepeatRule.DAY_IN_MONTH,
cal.get(Calendar.DAY_OF_MONTH));
event.setRepeat(rule);
```

This makes the Event available in this year's calendar, and repeatedly on the same day of the month for an unlimited number of years. The final step is then to commit the Event (to save your Event on the device) and close the Event list (to release resources).

Section 5. Deploy and run on a Windows Mobile device

This section guides you through the steps required to install the latest WebSphere Everyplace Micro Environment runtime on your Windows Mobile device and also shows you how to deploy and run the BirthdayReminder MIDlet application.

Install J9 MIDP 2.0

Go to the location where you downloaded your WebSphere Everyplace Micro Environment 6.1 - CLDC 1.1/MIDP 2.0 for Windows Mobile 5.0. (You did this in the Prerequisites section.)

Run the installer and select a temporary folder for your installation.

After the installation process is complete, go to the temporary folder and extract the file `weme-wm50-arm-midp20_6.1.0.20060317-111429.zip`. Note that the numeric parts of this file name may change depending on the version.

Next, you should have the directories `"/bin"` and `"/lib"` among the ones extracted. Use ActiveSync to create a folder called "J9" in your Windows Mobile device and copy both directories (bin and lib) to your new J9 folder. The directory structure on your device should look like:

```
\J9
 \J9\bin
 \J9\lib
```

Install the PIM Optional Package

Navigate to the place where you installed the PIM Optional Package (back in the [Install MIDP2.0 and PIM Optional Package](#) section). If you do not remember, look at the "Edit JRE" window in Eclipse, as shown in the [Configure Eclipse to use J9](#) section.

You will need to copy two files to your device: `pim.jar` and `pimdll.dll`. Copy `pim.jar` to your device's `\J9\lib\jclMidp20\ext` folder and `pimdll.dll` to your device's `\J9\bin` folder.

Install the BirthdayReminder MIDlet

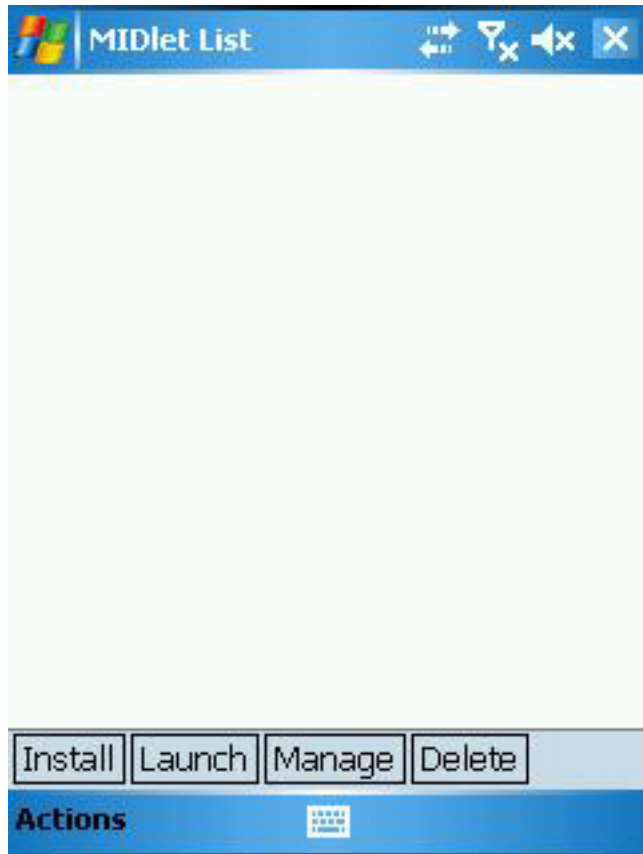
Download the `BirthdayReminderMidlet.zip` from the Downloads section. Extract the contents of the file to the root of your device. You should end up with the following files on the root of your Windows Mobile device: `\BirthdayReminder.jad` and `\BirthdayReminder.jar`.

The "jad" file describes the MIDlet contents. Inside the "jar" file, you will also find a "Manifest.mf" file, which also describes the MIDlet.

If, after this tutorial, you decide to create your own MIDlet using Eclipse, be sure to recreate both "jad" and "manifest" files for your own MIDlet. To do that, just edit the existing ones and save with a different name.

Next, using the File Explorer on your device, navigate to `\J9\bin` and run `emulator.exe`. This runs the AMS MIDlet manager from where you can install and manage your MIDlets, as [Figure 19](#) shows.

Figure 19. AMS MIDlet list



Inside AMS, click **Install** and enter the URL: file:\\\\BirthdayReminder.jar.

Figure 20. AMS MIDlet installation



Click **OK** and the MIDlet will install. During the installation process, you will be asked for permission to install the untrusted MIDlet because it has not been signed; say Yes to proceed with the installation. After the installation process is complete, you will see the BirthdayReminder MIDlet in the AMS's MIDlet list.

Run the MIDlet

1. From AMS, select the BirthdayReminder MIDlet and click **Launch**.

Figure 21. AMS MIDlet launch



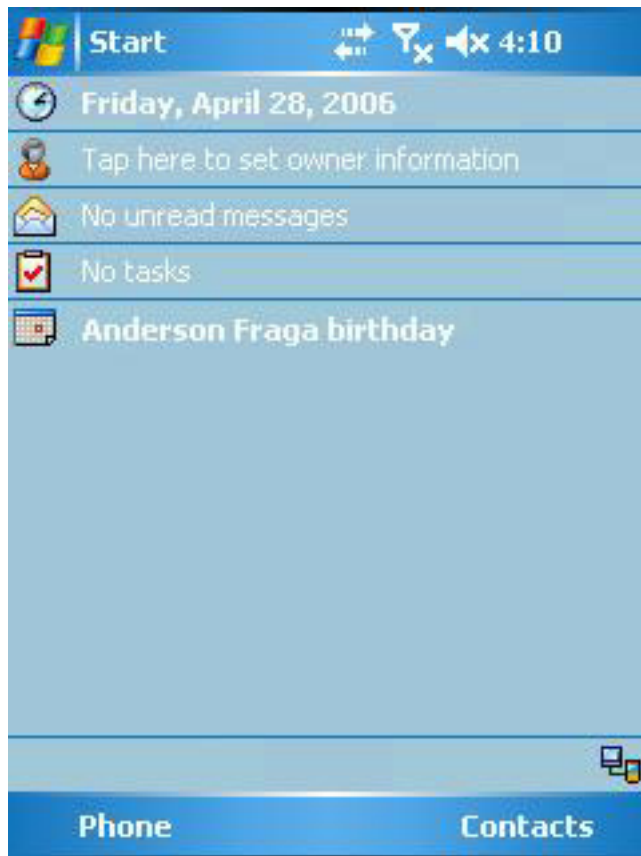
2. Type your friend's name and birthday.
Figure 22. BirthdayReminder MIDlet



The screenshot shows a mobile application window titled "Birthday Reminder". It features three input fields: "Last Name:" containing "Fraga", "First Name:" containing "Anderson", and "Brithday:" containing a date picker set to "4/28/2006". At the bottom of the window, there is an "OK" button and an "Actions" bar with a keyboard icon.

3. Click **OK** to add the birthday as a calendar event. As a security feature of the PIM API, you will be notified that the MIDlet wants to access your calendar events. Say Yes to allow the MIDlet to save the data. After this, when the birthday approaches, you will see a notification in your Windows Mobile main screen and calendar.

Figure 23. BirthdayReminder notification



Section 6. Summary

In this tutorial, I've covered all the steps required to get you started with J9 and the JSR-75 Personal Information Management API. Starting with how to configure your development environment, you learned how to configure the Eclipse SDK to develop a J9 PIM application.

Next, I covered the core components of the PIM API with use of code snippets to demonstrate the practical usage of the APIs. With your development environment configured and with a better understanding of the PIM APIs, you developed your first PIM MIDlet application.

Now you have all the tools and know-how to develop PIM applications that can help you keep your personal data organized and easy to access. So go ahead and explore new PIM application ideas.

Downloads

Description	Name	Size	Download method
Source code	BirthdayReminder.zip	6KB	HTTP

[Information about download methods](#)

Resources

Learn

- [JSR-75 PIM Optional Package Specification](#): Find the JSR-75 PIM Optional Package Specification on the JSR 75: PDA Optional Packages for the J2ME Platform Web site.
- [JSR 118: Mobile Information Device Profile 2.0](#): Locate the MIDP 2.0 specification on the JSR 118: Mobile Information Device Profile 2.0 Web site.
- [JSRs: Java Specification Requests](#): You can find the complete list of optional packages on the JSRs: Java Specification Requests Web site.

Get products and technologies

- [Eclipse Foundation](#): Get the Eclipse SDK on the Eclipse Foundation Web site.
- [WebSphere Everyplace Micro Environment](#): The latest version of IBM's WebSphere Everyplace Micro Environment runtimes are on the WebSphere Everyplace Micro Environment Trial and Evaluation Runtimes Web site.

Discuss

- [developerWorks blogs](#): Participate in the developerWorks community.

About the author

Anderson Fraga



Anderson Fraga is a software engineer at IBM Canada. He is a member of the J9 Java Class Libraries team and works with more platforms than he could ever imagine. You can contact Anderson at anderson_fraga@ca.ibm.com.