Eclipse ME
中文文档

目录

1. 概述

EclipseME 是帮助你开发 J2ME MIDlet 的 Eclipse 插件。EclipseME 可以帮助你简化 Java 2 Micro Edition (C) Midlet 的开发工作。 EclipseME 帮助你轻松的把无线工具包整合到 Eclipse 开发环境中， 使你不必再担心 J2ME 开发有何特殊需求，而可以把所有精力集中在应用开发上。 EclipseME 提供了下列功能以帮助开发者轻松的创建 Midlets：

- 无线工具包首选项
- 创建新的 J2ME Midlet 套件项目
- 创建新的 Midlet
- Java 应用描述文件(JAD)编辑器
- 对 class 文件进行增量的/自动的预检验
- 支持从 Eclipse 中启动运行 J2ME 模拟器
- 打包以及部署（包括无线下载方式）测试， 并可以使用ProGuard对输出包进行混淆

EclipseME 项目主页： http://www.eclipseme.org/

2. 先决条件

下面就是安装本插件的先决条件:

- Sun Java2 1.4.x
- Eclipse 集成开发环境
- EclipseME 1.1.0 版以及更新的版本需要 Eclipse 3.1。新版本已经放弃了对 Eclipse 3.0 的支持
- EclipseME 1.0.0 版可以同时支持 Eclipse 3.0 或 Eclipse 3.1
- 1.0.0 以前版本的 EclipseME 只支持 Eclipse 3.0

本文档是在 Windows XP 上运行的 Eclipse 3.1 上整理的。

下列项目则是可选的:

- ProGuard混淆器 http://proguard.sourceforge.net/
  如果你希望使用本插件生成混淆的Jar包，则必须安装ProGuard。

2.1. 本插件支持的无线工具包

支持的工具包列表，目前，经过测试，EclipseME 支持下列无线工具包。 点击下面相应链接以获得进一步信息。

- Sun 无线工具包
- 摩托罗拉 J2ME™ SDK
- 诺基亚 J2ME 开发者套件 2.2
- 索尼爱立信 J2ME SDK
- Sprint PCS Java 无线工具包
- 西门子 60 系列 SMTK

图例

1. 经测试可正常工作
2. 已获得可正常工作的报告
3. 可以部分的工作
4. 无法工作

其它工具包可能也可以和 EclipseME 一起工作，但我们不会对它们进行支持和测试。 特别的，从 0.6.0 版本开始，EclipseME 开始支持符合“统一模拟器接口”(UEI)规范的模拟器。因此，虽然有些工具包未出现在下面列表中，但仍有可能被当作“普通 UEI 模拟器”而正常工作。 请参考安装说明中的为插件配置无线工具包一节 以获取配置 UEI 模拟器的进一步信息。

- Sun无线工具包
  http://wireless.java.sun.com.

  - Windows
    - Sun's 1.04 无线工具包
    - Sun's 2.0 无线工具包
    - Sun's 2.1 无线工具包
    - Sun's 2.2 无线工具包
    - Sun's 2.3 无线工具包
  - Linux
    - Sun's 2.1 无线工具包
    - Sun's 2.2 无线工具包
    - Sun's 2.3 无线工具包

- 摩托罗拉J2ME™ SDK
  http://www.motocoders.com/.
  - Windows
    - 版本 4.1 版
      模拟器 7.2 - 可以编译，但不能调试和运行
      模拟器 7.5 - 可以编译，但不能调试和运行
      模拟器 A.1 - 可以“运行”和“调试”
      模拟器 M.1 - 可以编译，但不能调试和运行
    - 版本 4.3
      模拟器 A.1 - 可以“运行”和“调试”
      模拟器 A.3 - 仅“运行”
      模拟器 M.1 - 可以“运行”和“调试”
      模拟器 M.3 - 可以“运行”和“调试”
    - 版本 4.4
      模拟器 A.1 - 可以“运行”和“调试”
      模拟器 A.3 - 仅“运行”
      模拟器 M.1 - 可以“运行”和“调试”
      模拟器 M.3 - 可以“运行”和“调试”
    - 版本 5.2
      模拟器 A.1 - 经调整可以“运行”和“调试”(参见下面的注解 1)
      模拟器 A.3 - 可以编译，但目前无法“运行”和“调试”
      模拟器 A.4 - 可以编译，但目前无法“运行”和“调试”
      模拟器 M.1 - 经调整可以“运行”和“调试”(参见下面的注解 2)
      模拟器 M.3 - 可以“运行”和“调试”
  - Linux
    不可用

注解 1: Motorola SDK 5.1.2 版和 5.2 版缺失了一个名为"dbgclasses.zip"的文件，但 A.1 模拟器却需要这个文件。 发布版无法正常调试，因为当遇到断点的时候正好要用到这些缺失的类。 这个文件一般应位于 EmulatorA.1/bin 目录下，你可以从 4.4 版本的 SDK 中复制过来。 经过这样修正，调试就应该可以正常进行了。

注解 2: Motorola M.1 模拟器并未提供（至少文档中并未提及）从 bin/resources 目录选择设备皮肤的运行参数。 而 4.x 版系列模拟器只有一个皮肤 - A760。到 5.x 版系列则改成了 A760_A760i.props 和 A768_A768i.props 这两个皮肤， 但是模拟器在默认情况下仍然会

试图寻找一个叫 A760.props 的文件，而此文件实际并不存在。因此只要把 A760_A760i.props 和 A768_A768i.props 这两个文件之一复制一份成 A760.props（或重命名为 A760.props），然后就都可以正常工作了。

- 诺基亚J2ME开发者套件 2.2
  http://www.forum.nokia.com.
  - Windows
    - Nokia_6230_MIDP_Concept_SDK_Beta_0_2
    - Nokia S40 DP20 SDK 1.0
    - Series 60 MIDP SDK 2.1 Beta
    - Nokia 7210 MIDP SDK v1.0
    - Series 90 MIDP Concept SDK Beta 0.1
  - Linux
    - 绝大部分模拟器可以正常工作
  - 注意
    - 40 系列 SDK 仅支持“运行”。
    - 60 系列和 90 系列 MIDP SDK 的性能较差，因此必须延长调试器的超时设定。
    - 调试时在一些情况下 Eclipse 会抛出 IllegalArgumentException 异常。

- 索尼爱立信J2ME SDK
  http://www.sonyericsson.com/developer
  - Windows
    - 版本 2.1.3
    - 版本 2.2-2.2.3
  - Linux
    - 不可用

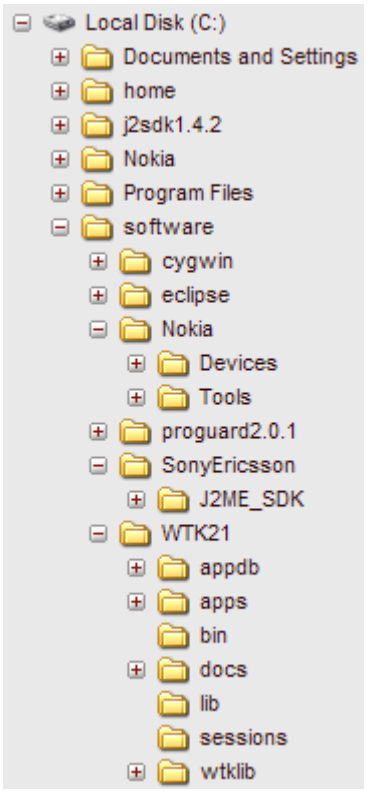
- Sprint PCS Java无线工具包
  http://developer.sprintpcs.com/site/global/home/p_home.jsp
  - Windows 版本
    - 2.0.13
  - Linux
    - 不可用

- 西门子 60 系列 SMTK
  - Windows
    - 版本 2.x - 可以“运行”，但仍不能调试
    - 版本 3.x - 可以“运行”，但仍不能调试
  - Linux
    - 不可用

关于工具包的导入
在任何情况下，添加新的平台组件时必须指定无线工具包的根目录。

普通 UEI 模拟器
UEI 模拟器的根目录是相对模拟器的执行文件而指定的。比如下面的例子，
C:\software\toolkits\mytoolkit\bin\emulator.exe
那么这个模拟器的根目录就应该是
C:\software\toolkits\mytoolkit

特定工具包
假设工具包是按照类似下面的目录结构安装的



那么这些工具包就应该指定下面的目录来进行添加：
C:\Program Files\Motorola\SDK v4.3 for J2ME
C:\software\Nokia
C:\software\SonyEricsson
C:\Program Files\Sprint PCS
C:\software\WTK21
剩下的工作 EclipseME 就会自动完成了。

3. 安装

本文档描述了如何安装和配置 EclipseME。 基本上就是三个步骤：安装无线工具包，安装 EclipseME 本身，配置 EclipseME 和 Eclipse。

3.1. 安装之前

请首先确认你已符合先决条件。

如果你安装了早于 0.5.0 版的 EclipseME，请在安装新版本前卸载掉旧版本。 另外，请阅读移植说明， 看看把使用旧版 EclipseME 创建的项目更新到新版需要哪些特定步骤。

如果你已经安装了 0.5.0 或更新的版本，那么无需卸载它。接下来的安装流程会自动更新你的 EclipseME。

最后，如果你在没有 EclipseME 的情况下使用 Eclipse 和某些无线工具包开发过 J2ME 程序， 那么可能需要对你的 Eclipse 项目的 classpath 进行一些调整， 具体做法请参见把现存项目转换为 EclipseME 项目一节。

安装无线工具包

你可以安装任何你希望使用的无线工具包。 安装的时候，请记住安装后的工具根目录，因为在配置 EclipseME 的时候你需要这个信息。

对 Unix 用户，如果要在你的系统为多位用户安装无线开发包， 那么请确保这些用户对工具的内容拥有恰当的读和执行权限。 比如，在 EclipseME 的构建过程中，就需要访问 WTK 内含的预检器(preverifier)。 如果用户对预检器没有执行权限，那么构建就会失败。 同样的，如果用户对模拟器没有执行权限，那么他（她）就无法进行测试。

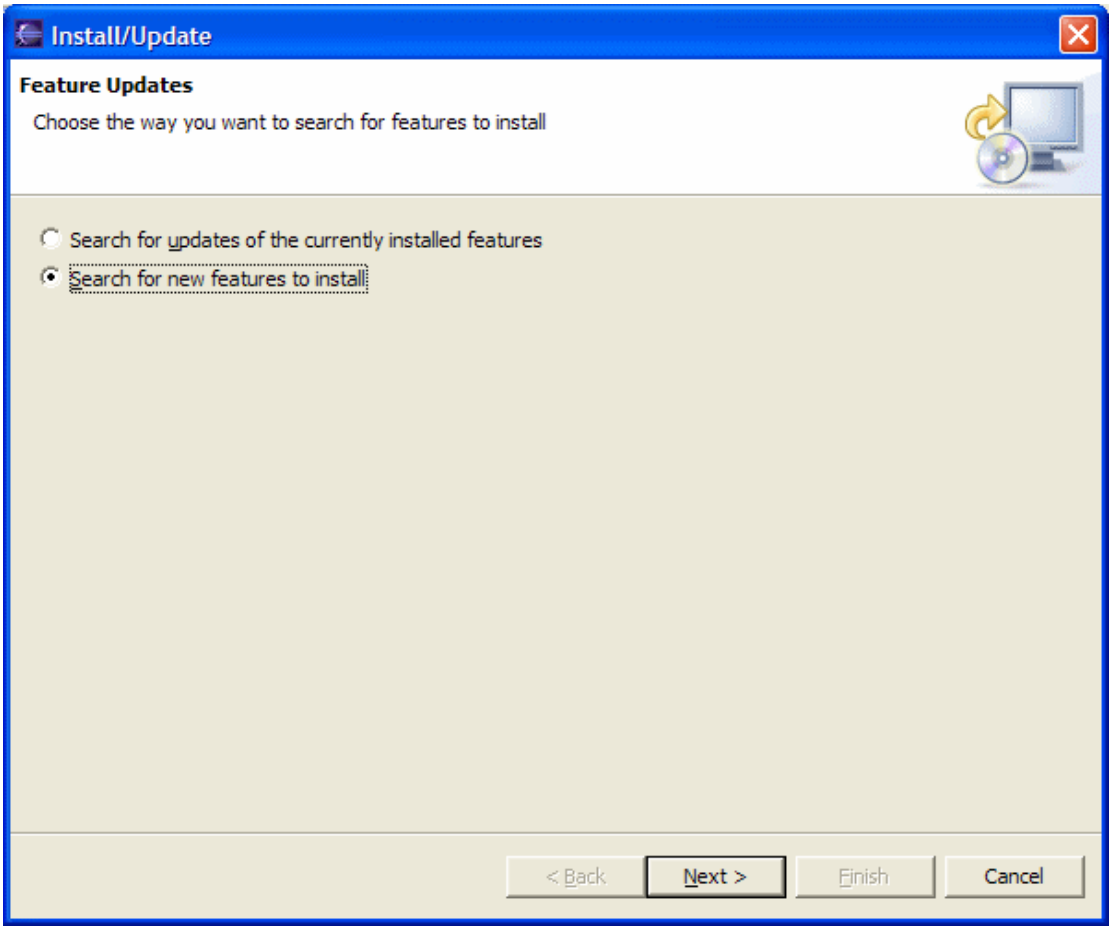
完成了上面这些步骤，你就可以安装 EclipseME 插件了。

3.2. 安装 EclipseME 本身

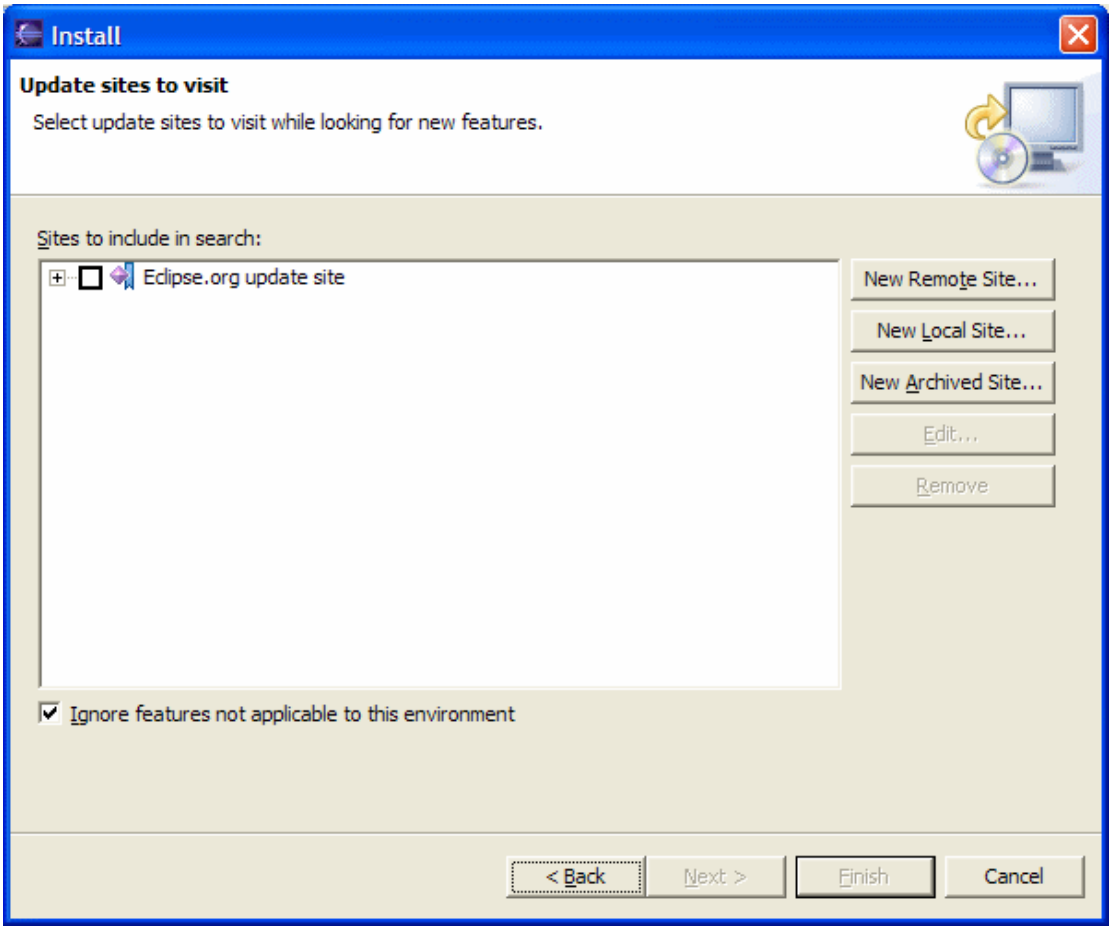
决定使用何种安装方法 0.5.0 版本和更早的版本只需要把发布包简单的解压到 Eclipse 的安装目录或插件目录即可。 但这已经不再是正确的安装方式。 从 0.5.5 版本开始，EclipseME 开始使用 Eclipse 的“更新站点”方式进行发布。 请按照下面的说明进行安装。

如果你想更新 EclipseME，你同样需要遵循下面的流程。 安装和更新的步骤是一样的。EclipseME 有两种基本的安装/更新方式：

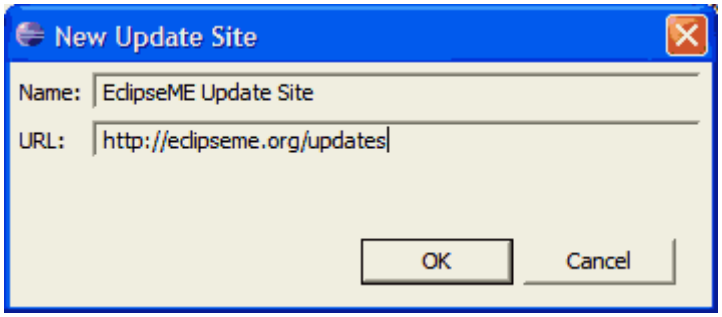
1. 使用 Eclipse 内建的站点更新功能来直接连接到 EclipseME 的更新站点进行安装/更新。 也可以手动下载一个完整安装包，并使用它进行安装。
2. 使用 EclipseME 更新站点进行安装从 Eclipse 的帮助菜单，选择 软件更新菜单项，然后选择 查找并安装...子项。 这时你会看到下面的对话框：



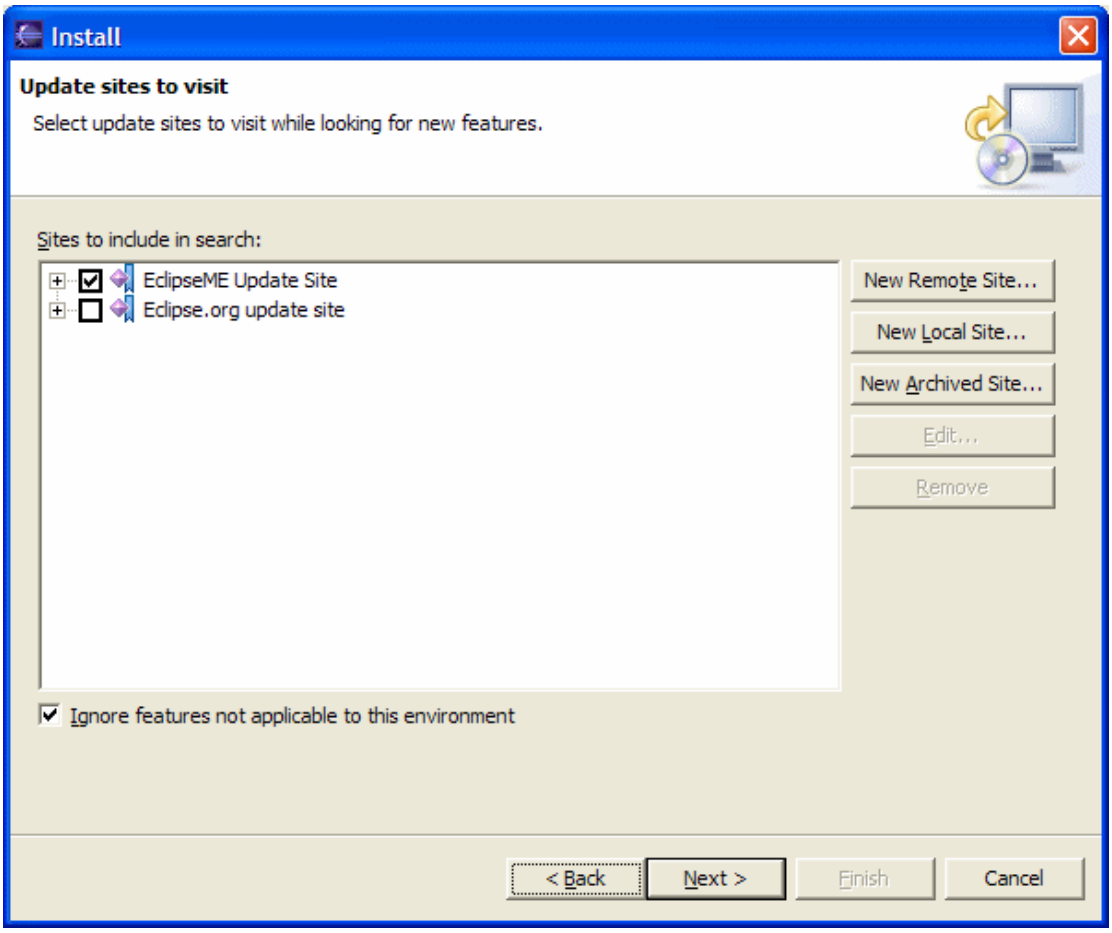
选中搜索要安装的新功能部件单选按钮，然后按下一步按钮。接下来你会看到下面的对话框：



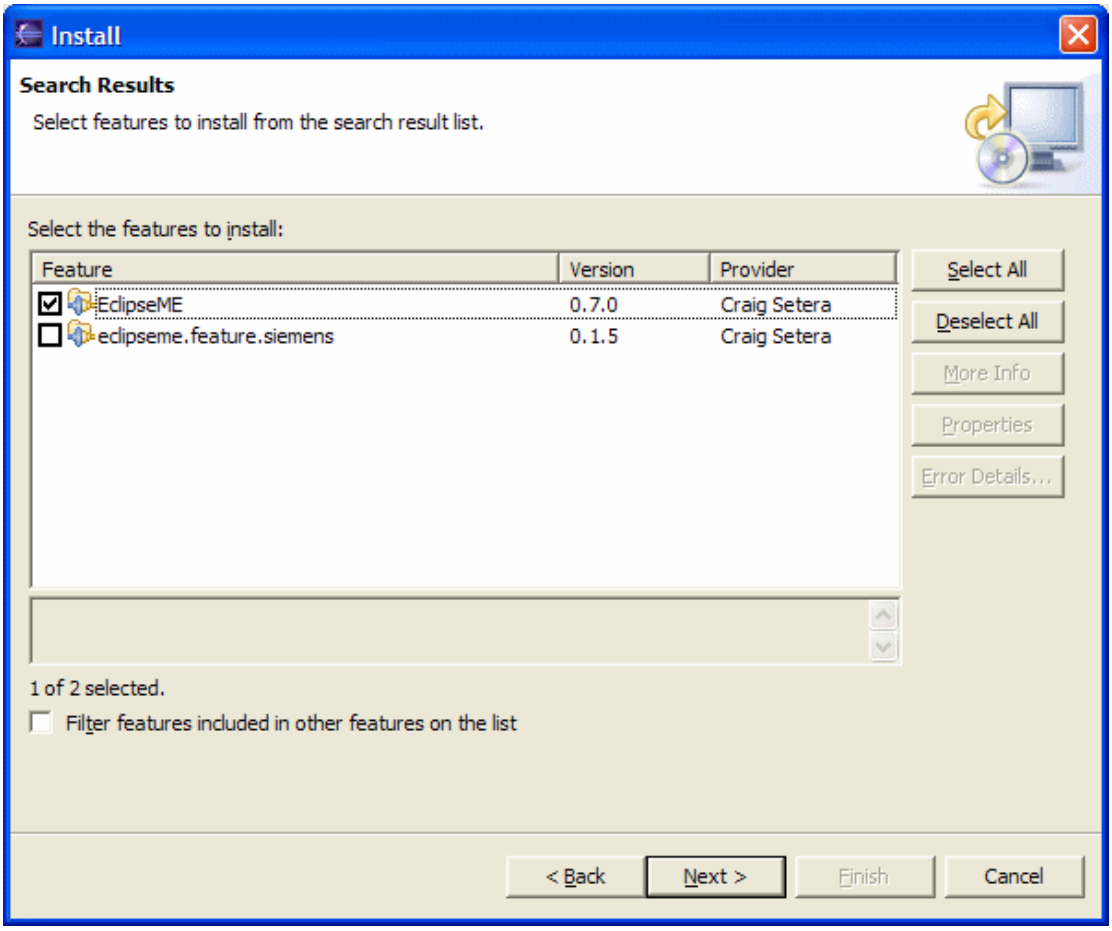
按新建远程站点...按钮。 在出现的新建更新站点对话框中输入更新站点的名字和地址 http://www.eclipseme.org/updates/ 然后按确定。



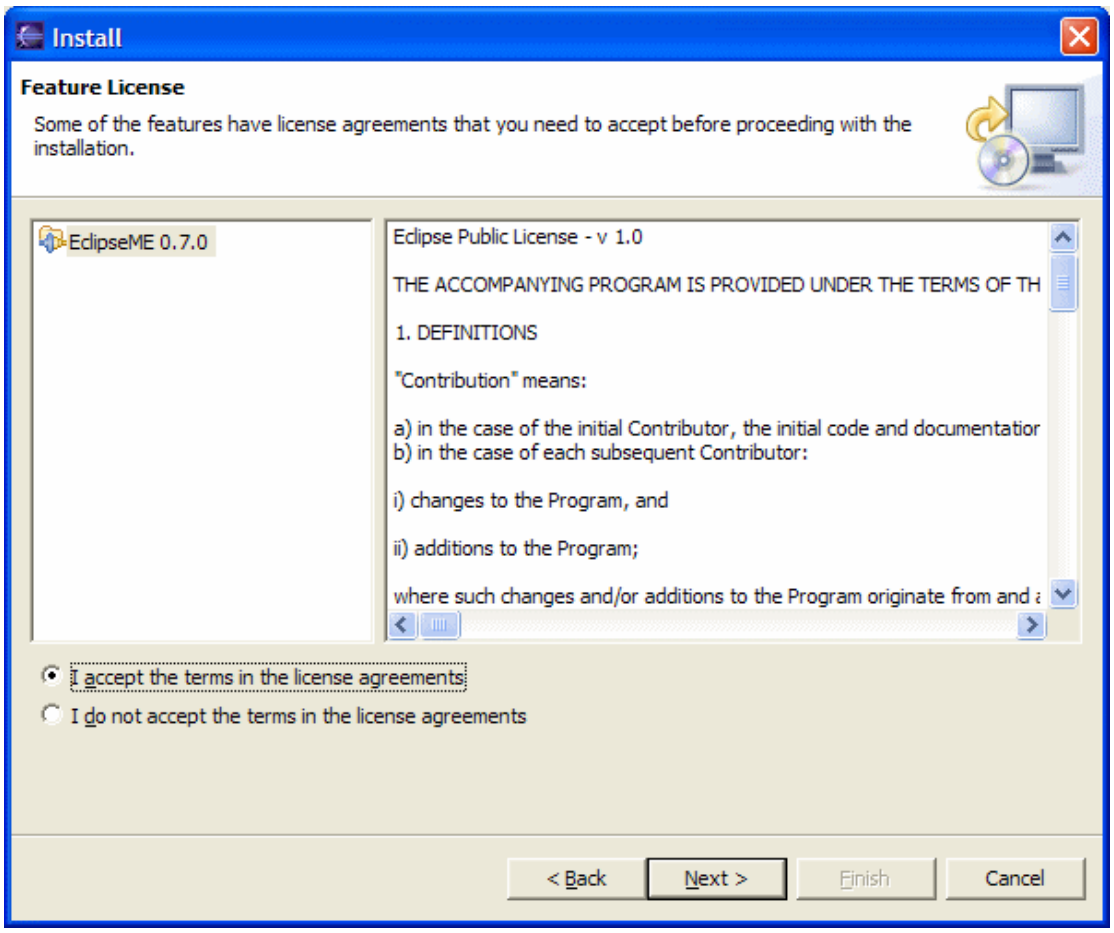
这时更新站点就出现在安装对话框的列表中了。Install dialog.



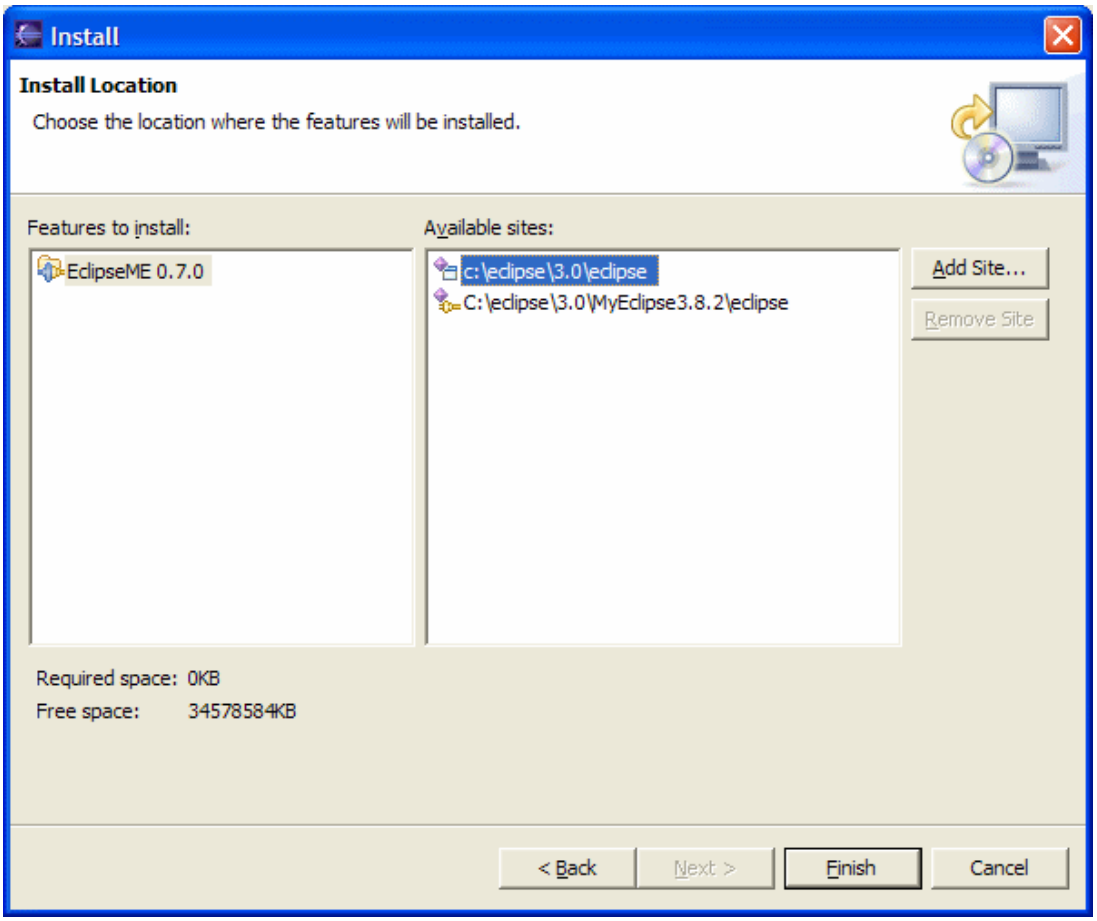
勾选"EclipseME update site"左边的复选框，然后按下一步。接下来你会看到下面的对话框：



勾选 EclipseME 左边的复选框。 如果希望同时安装其它的附加功能，就选中它们左边的复选框。 按下一步。 接下来会显示 EclipseME 的许可协议。



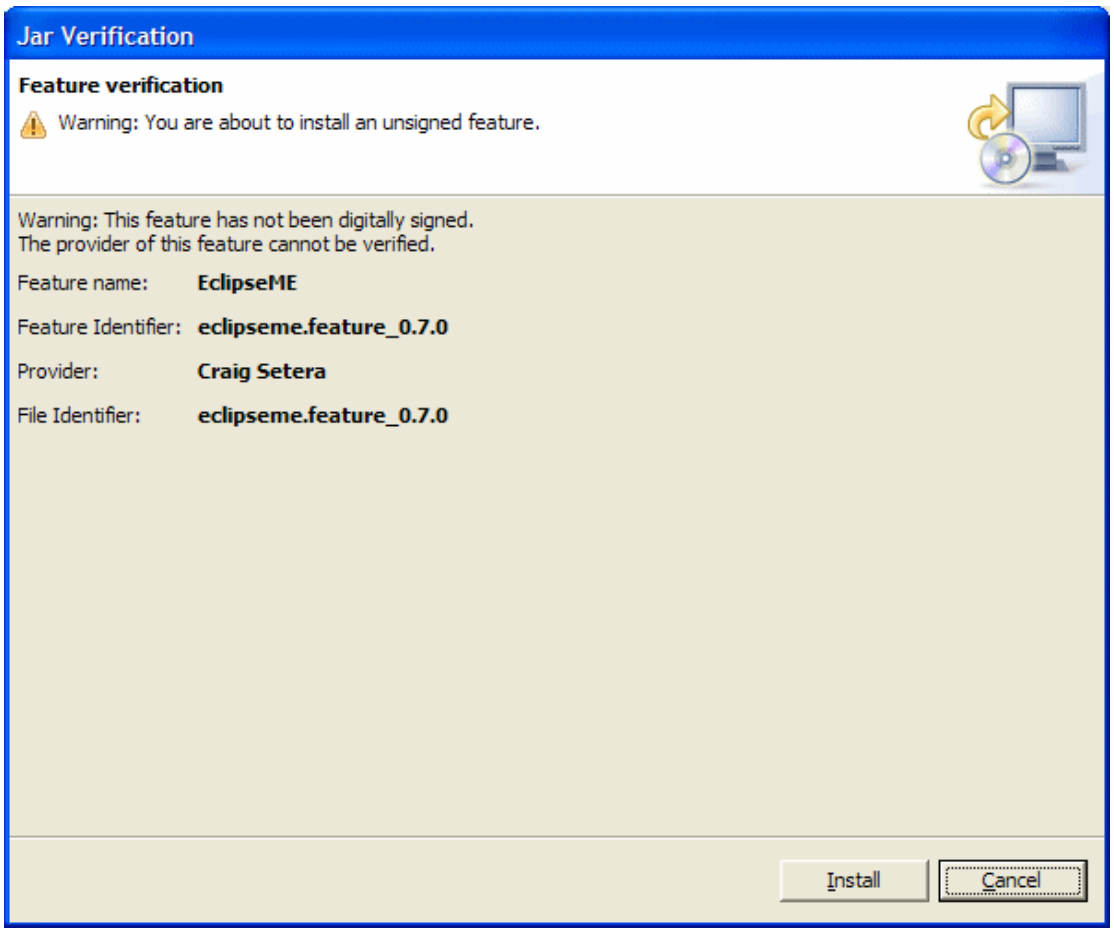
选择我接受许可协议中的所有条款然后按下一步。 接下来的对话框显示了 EclipseME 可以被安装到的位置。



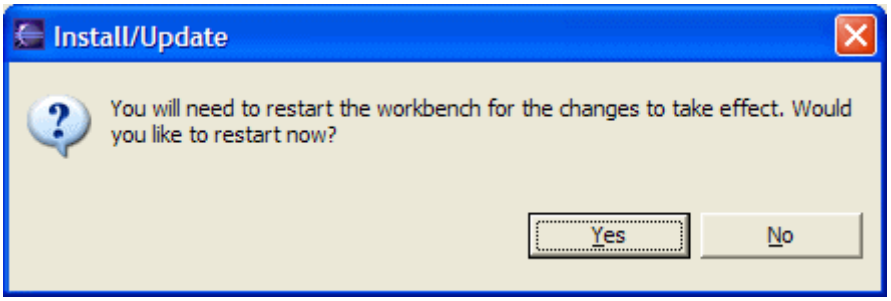
大部分情况下，只会列出唯一的一个位置，也就是 Eclipse 的安装目录。 但如果你安装了某些插件或功能部件，你也有可能看到其它的位置。

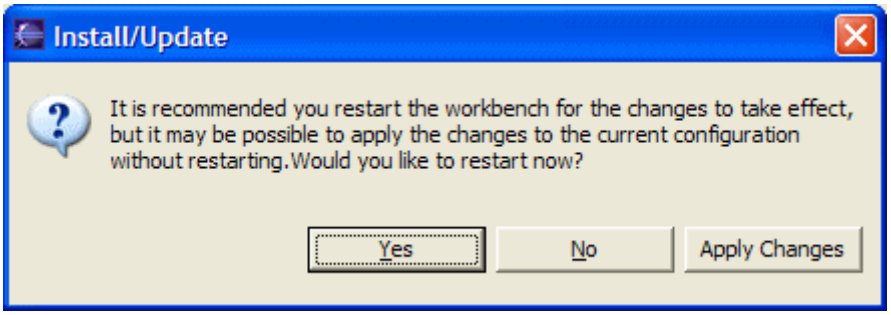
尽管你可以把 EclipseME 安装在别处，但我们建议你把它安装到 Eclipse 的安装目录下。当你选定了一个位置以后，按完成。

目前 EclipseME 发布包未作数字签名。 （维持一个 JAR 包的数字签名每年要花费 400 多美金。如果有谁愿意为 EclipseME 捐助这笔钱， 那我们很乐意为 JAR 包作数字签名。） 因此，接下来你会看到下面的警告窗口：



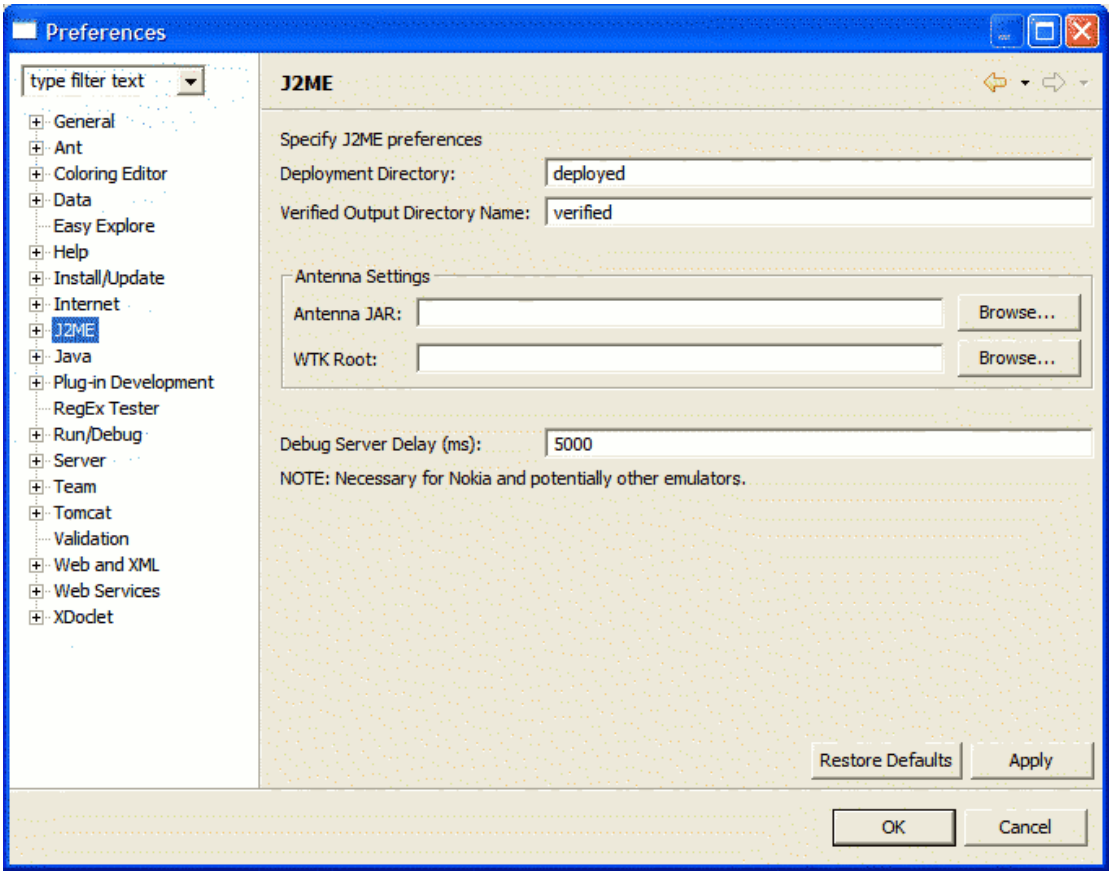
如果你确信你获得的 EclipseME 发布包来自一个著名站点（比如直接从 SourceForge.net 下载的），那么安装包的安全性应该是可以被信任的。如果你决定继续，那么按安装按钮。 这时候，Eclipse 就会开始从更新站点安装 EclipseME 了。安装结束的时候，你会看到下面两个对话框之一：



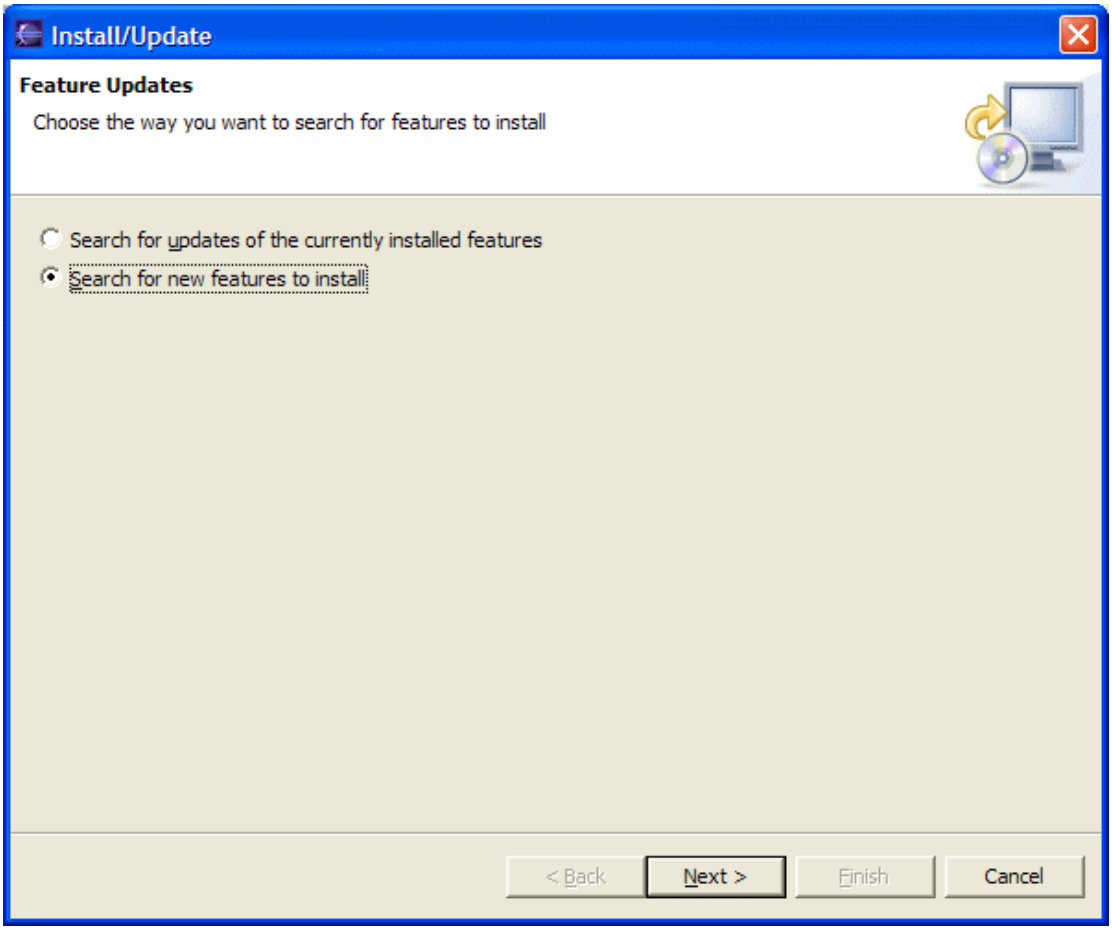


尽管那个新安装的对话框建议说可能不重启 Eclipse 也能继续，但你就干脆点儿重启 Eclipse 吧。 Eclipse 重启动之后，安装流程结束。

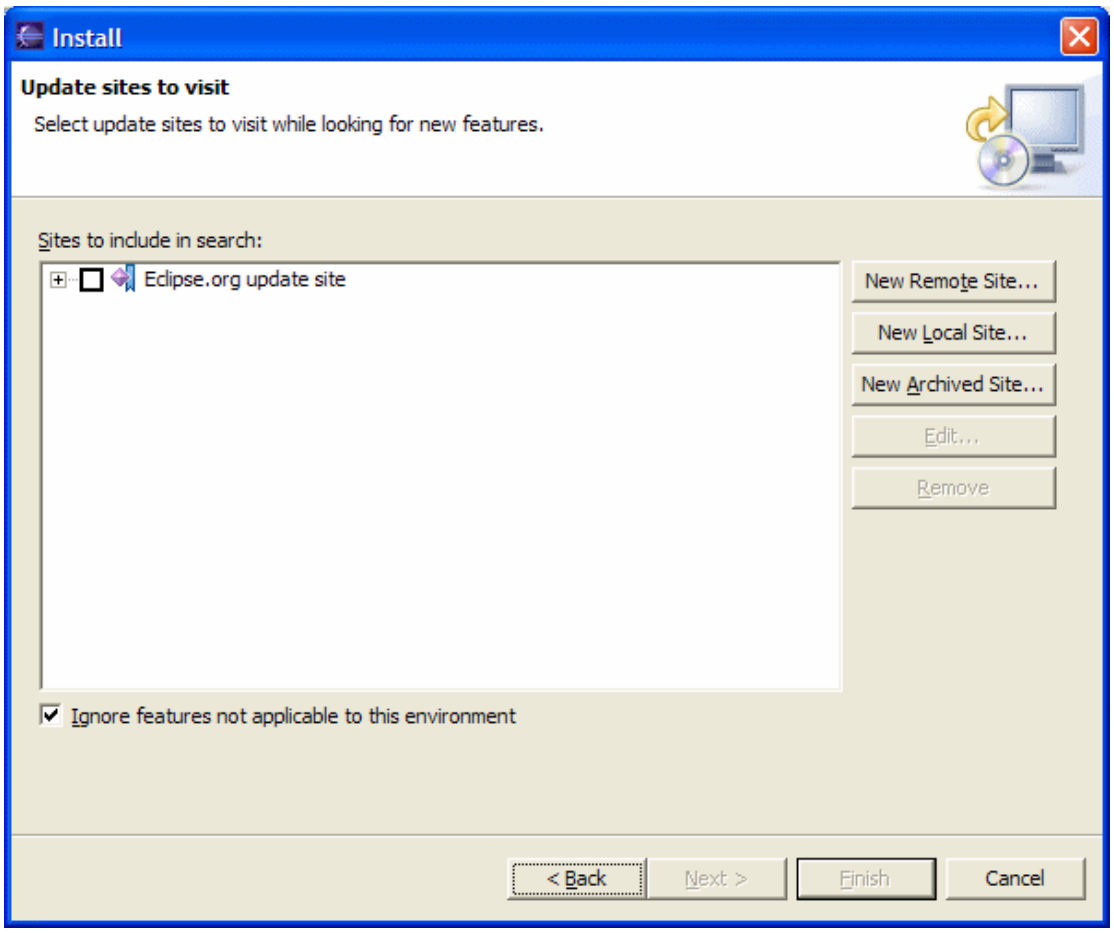
绝大多数情况下，上面的安装流程是可以正确更新旧版 EclipseME 的。
在极少数情况下，在 Eclipse 重启之后，EclipseME 可能仍未能正确“注册”到 Eclipse 中。在这种情况下，只要关闭 Eclipse，然后重启动 Eclipse 的时候增加一个 -clean 参数就可以了。 使用-clean 参数可以强制 Eclipse 重新扫描并更新它的所有插件信息。
如果正确安装了插件，那么在窗口/首选项对话框中会多出来一个 J2ME 选项分支。



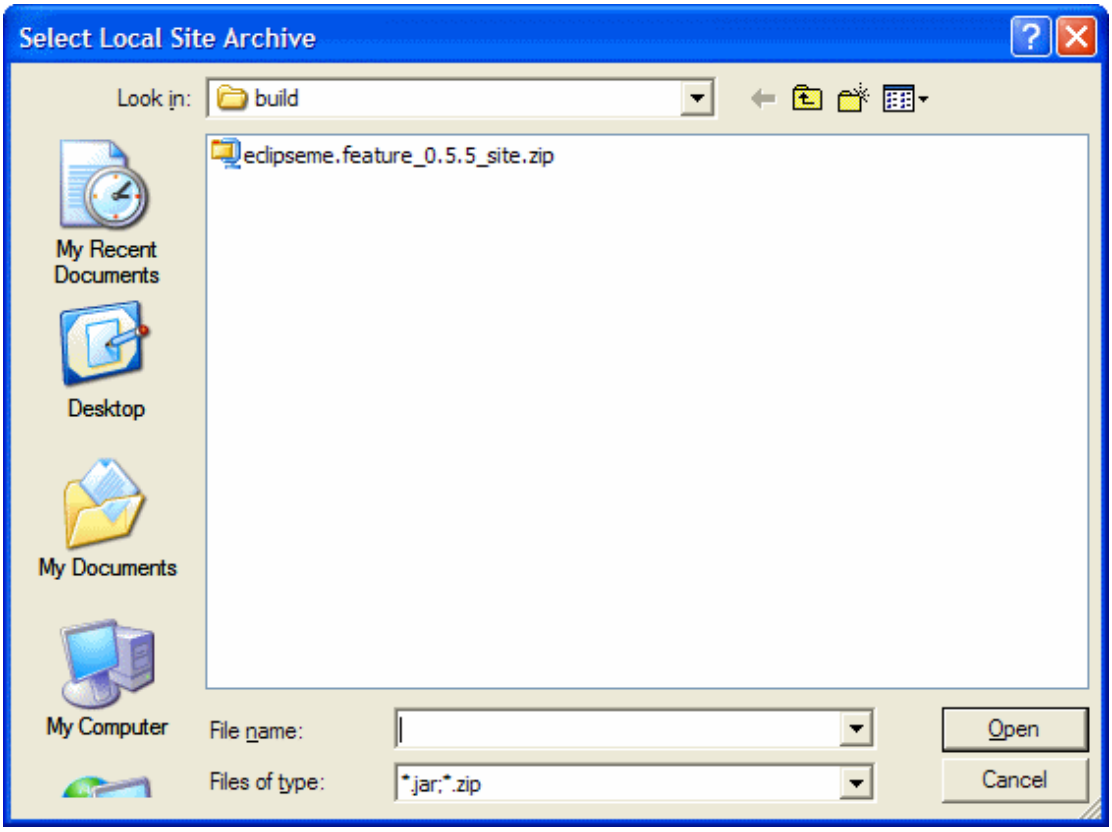
3. 使用下载的完整安装包进行安装
使用下载的安装包进行安装的流程和使用 EclipseME 更新站点进行安装的流程非常近似。
首先从EclipseME下载页面 下载最新的安装包。 从Eclipse的帮助菜单，选择 软件更新菜单项，然后选择 查找并安装...子项。 这时你会看到下面的对话框：



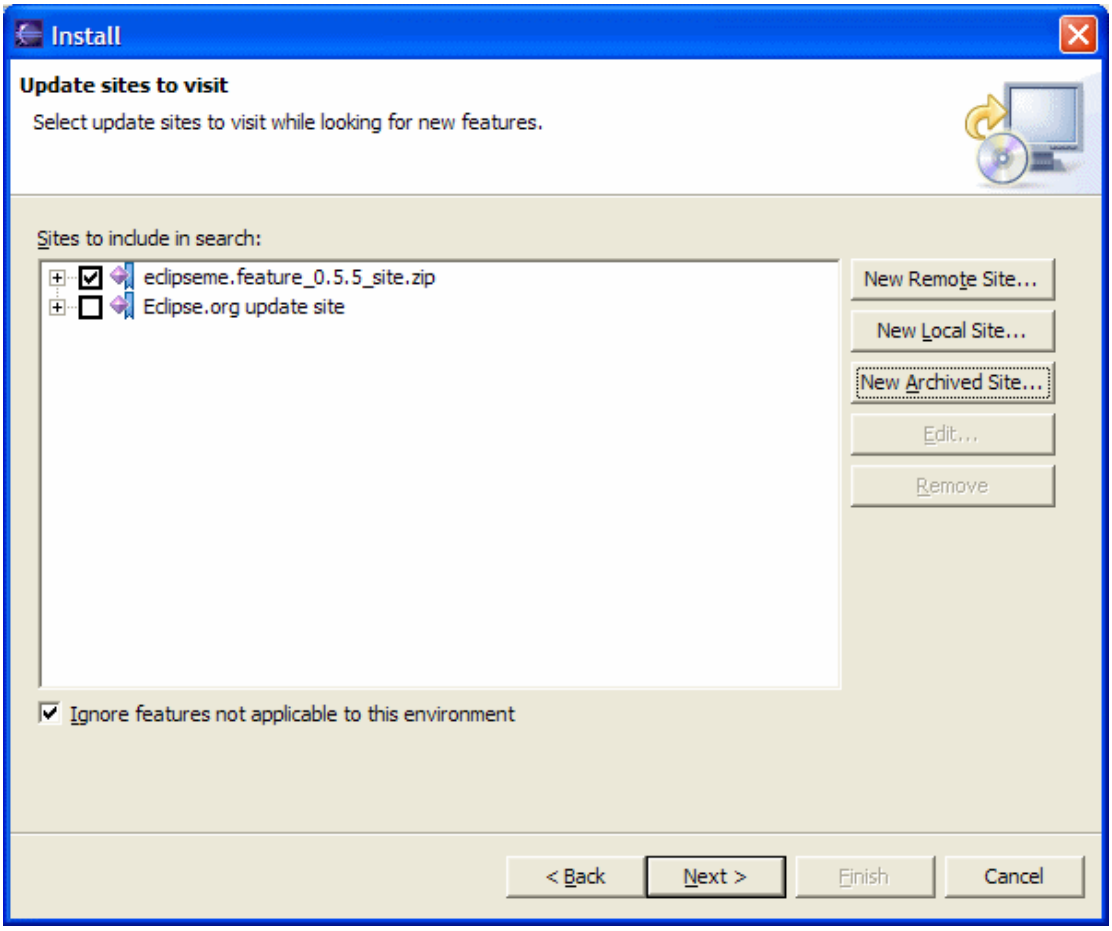
选中搜索要安装的新功能部件单选按钮 （即使你是想更新 EclipseME 也应该选择这个选项）。然后按下一步按钮。 接下来你会看到下面的对话框：



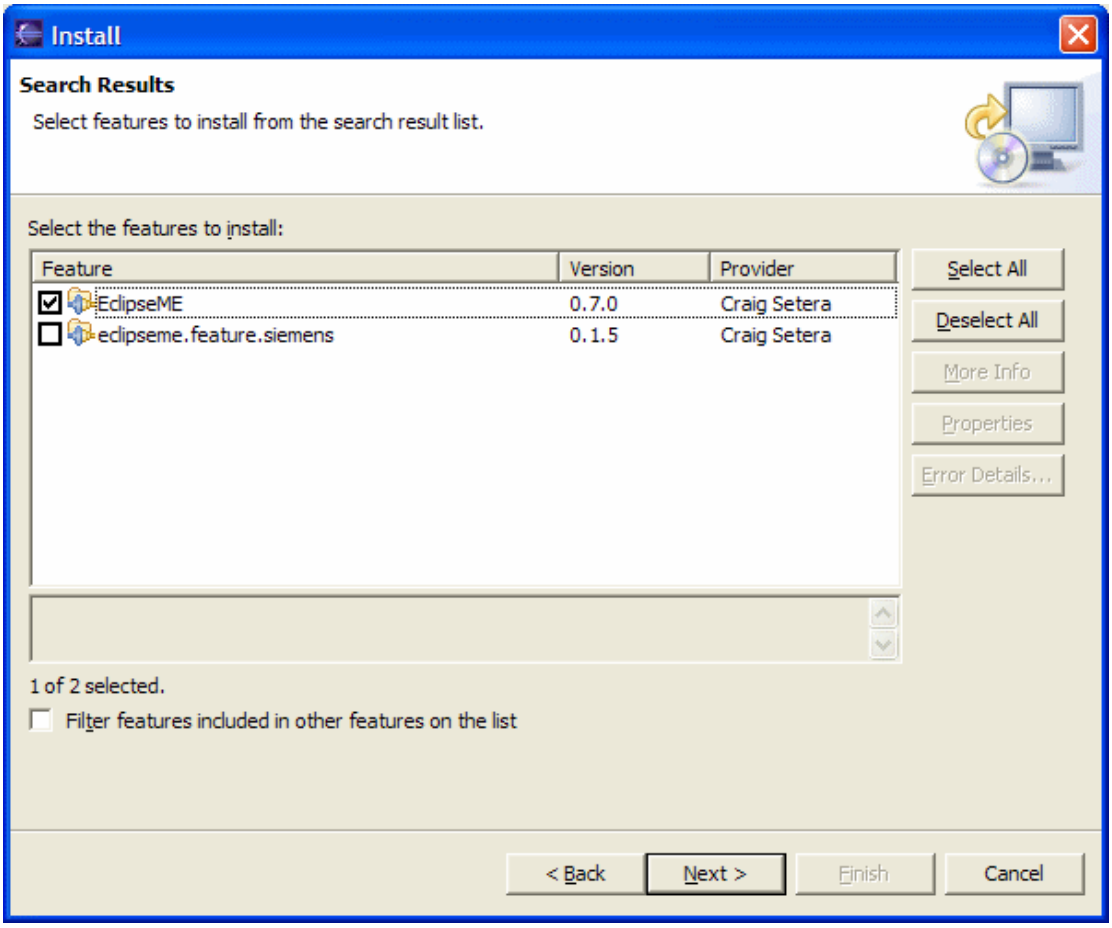
按新建已归档的站点...按钮。在接下来的选择本地站点归档对话框中， 选择你下载的 EclipseME 发布包 ZIP 文件然后按打开。



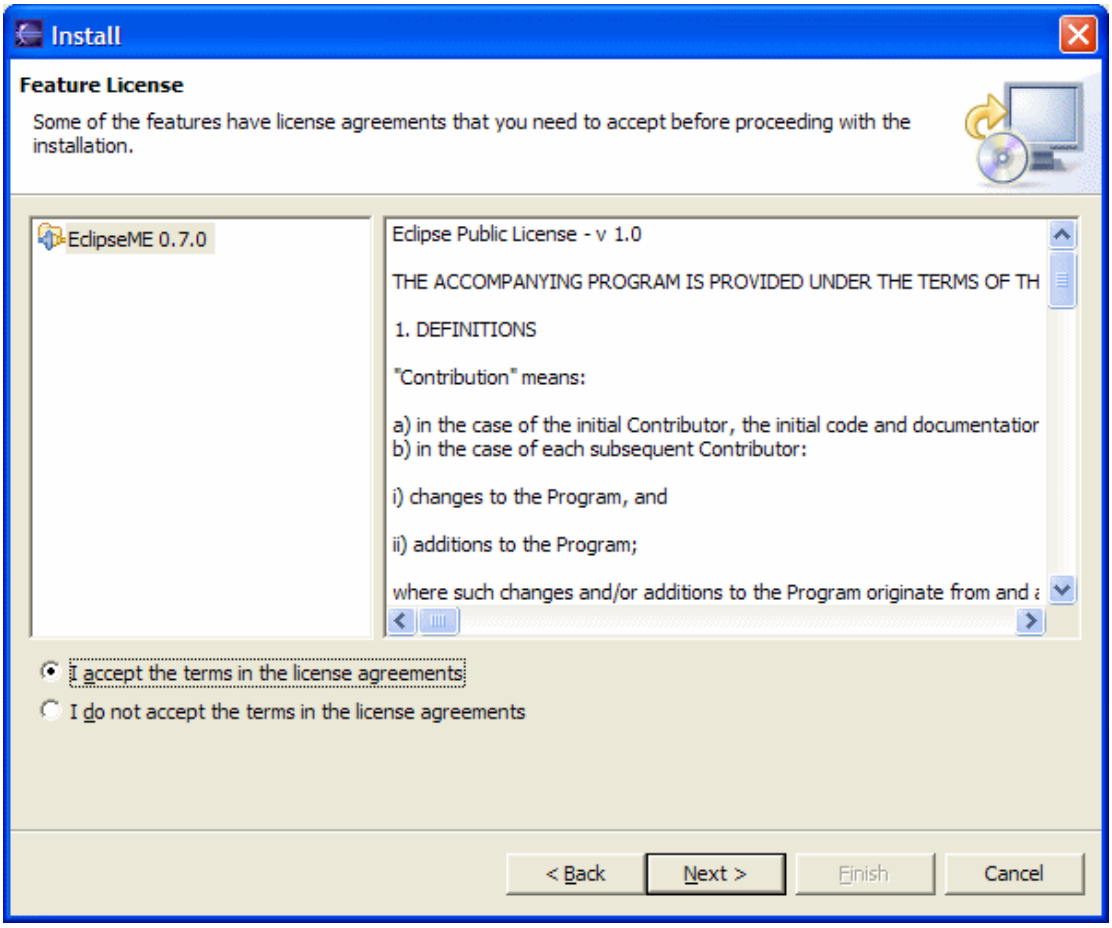
接着安装包文件会出现在安装对话框的列表中。



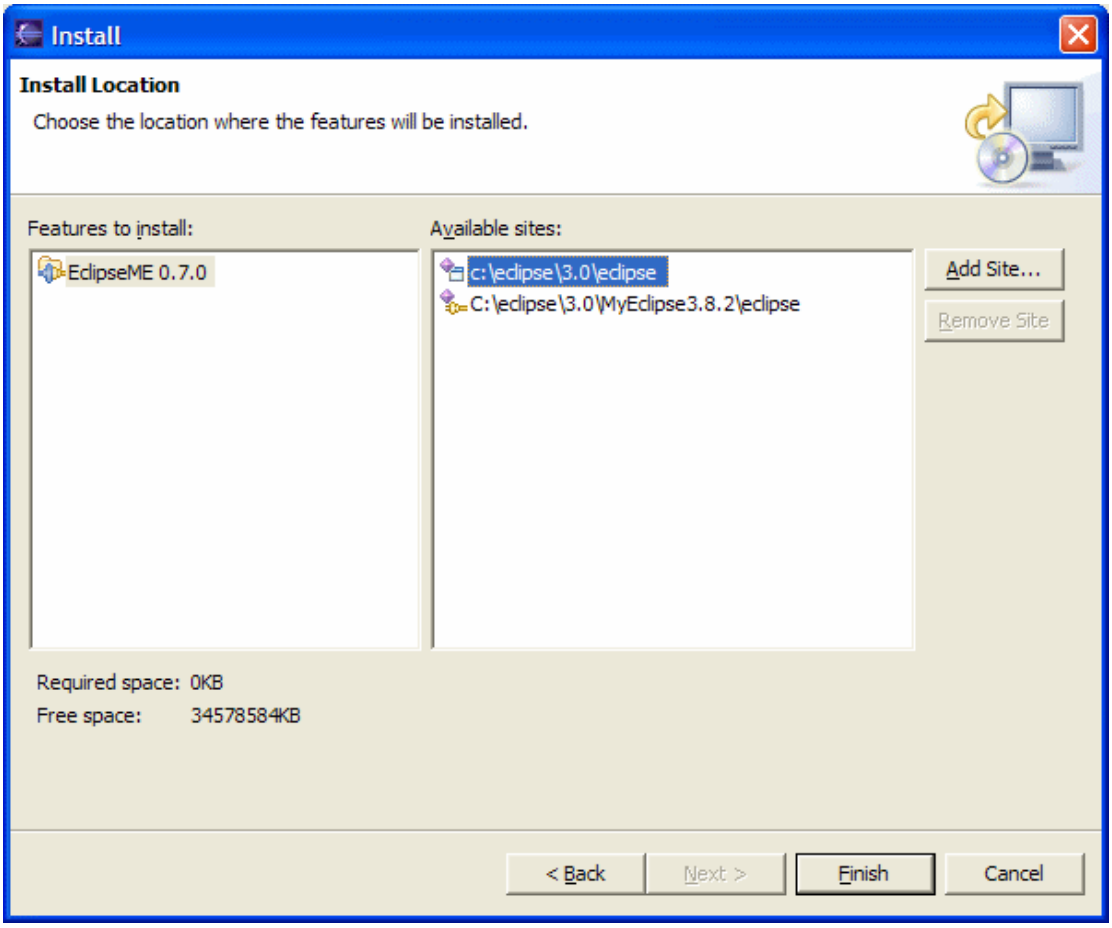
勾选安装包文件左边的复选框，然后按下一步。接下来你会看到下面的对话框：



勾选 EclipseME 左边的复选框。 如果希望同时安装其它的附加功能，就选中它们左边的复选框。 按下一步。接下来会显示 EclipseME 的许可协议。

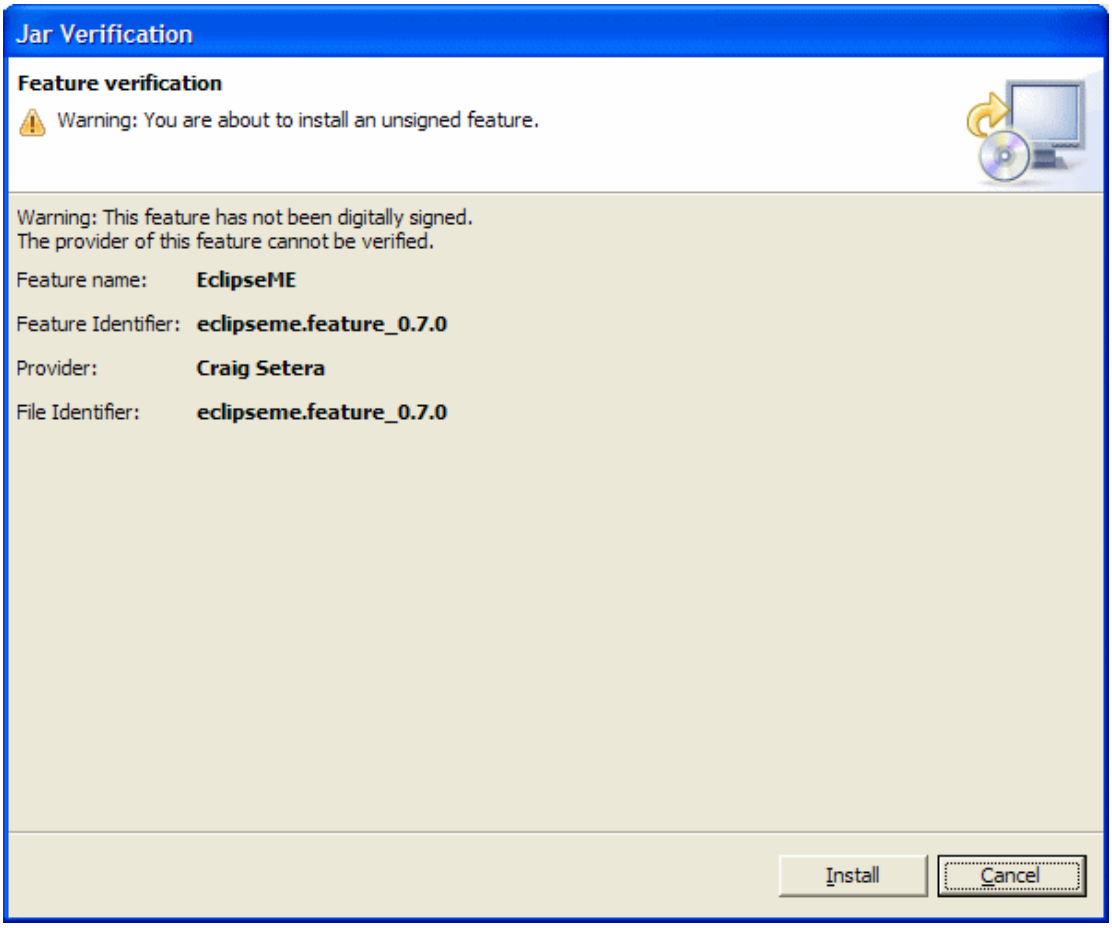


选择我接受许可协议中的所有条款然后按下一步。接下来的对话框显示了 EclipseME 可以被安装到的位置。

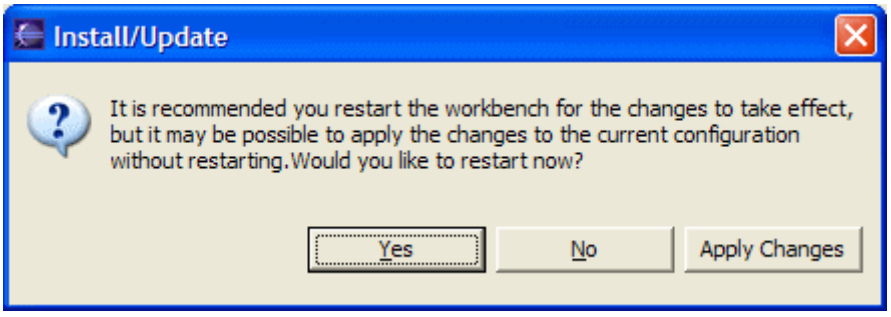


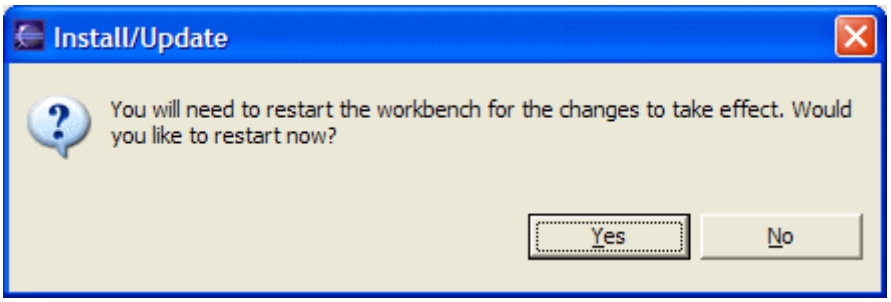
大部分情况下，只会列出唯一的一个位置，也就是 Eclipse 的安装目录。 但如果你安装了某些插件或功能部件，你也有可能看到其它的位置。尽管你可以把 EclipseME 安装在别处，但我们建议你把它安装到 Eclipse 的安装目录下。 当你选定了一个位置以后，按完成。

目前 EclipseME 发布包未作数字签名。(维持一个 JAR 包的数字签名每年要花费 400 多美金。如果有谁愿意为 EclipseME 捐助这笔钱， 那我们很乐意为 JAR 包作数字签名。) 因此，接下来你会看到下面的警告窗口：



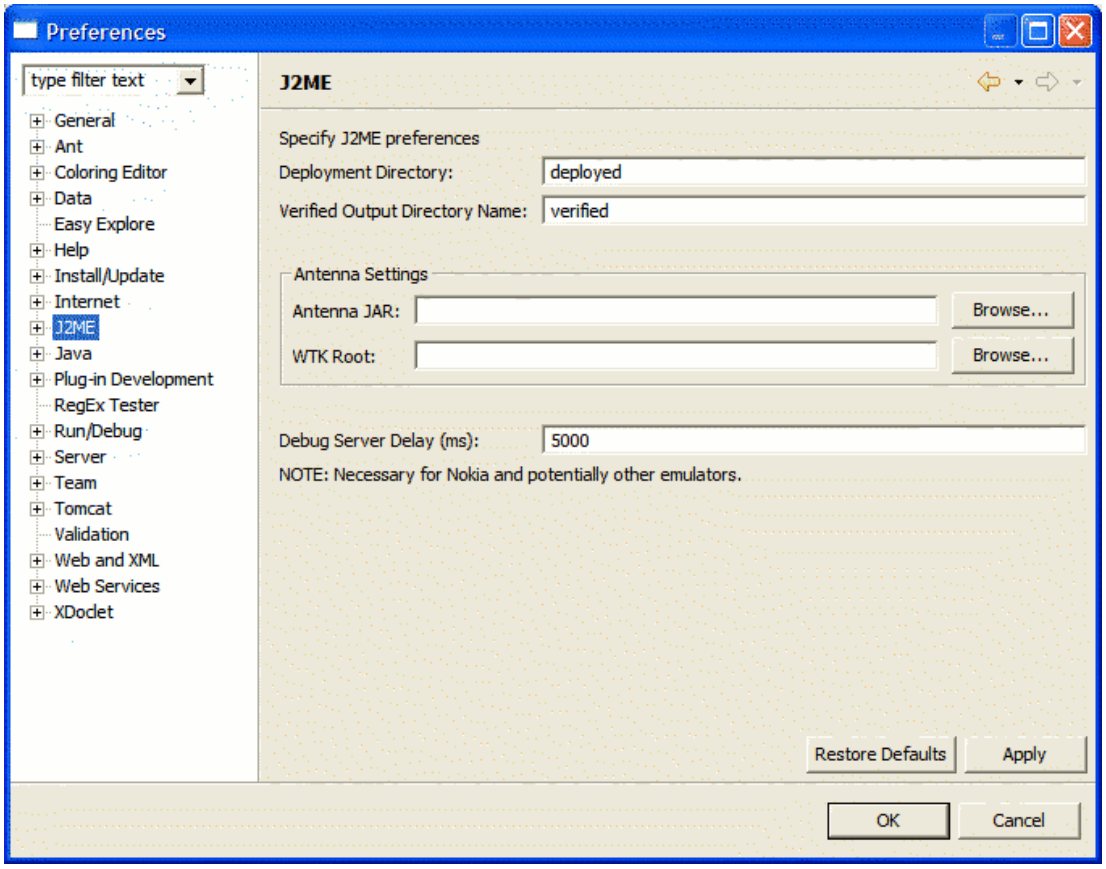
如果你确信你获得的 EclipseME 发布包来自一个著名站点（比如直接从 SourceForge.net 下载的），那么安装包的安全性应该是可以被信任的。如果你决定继续，那么按安装按钮。这时候，Eclipse 就会从安装包文件开始安装 EclipseME 了。安装结束的时候，你会看到下面两个对话框之一：





尽管那个新安装的对话框建议说可能不重启 Eclipse 也能继续，但你就干脆点儿重启 Eclipse 吧。

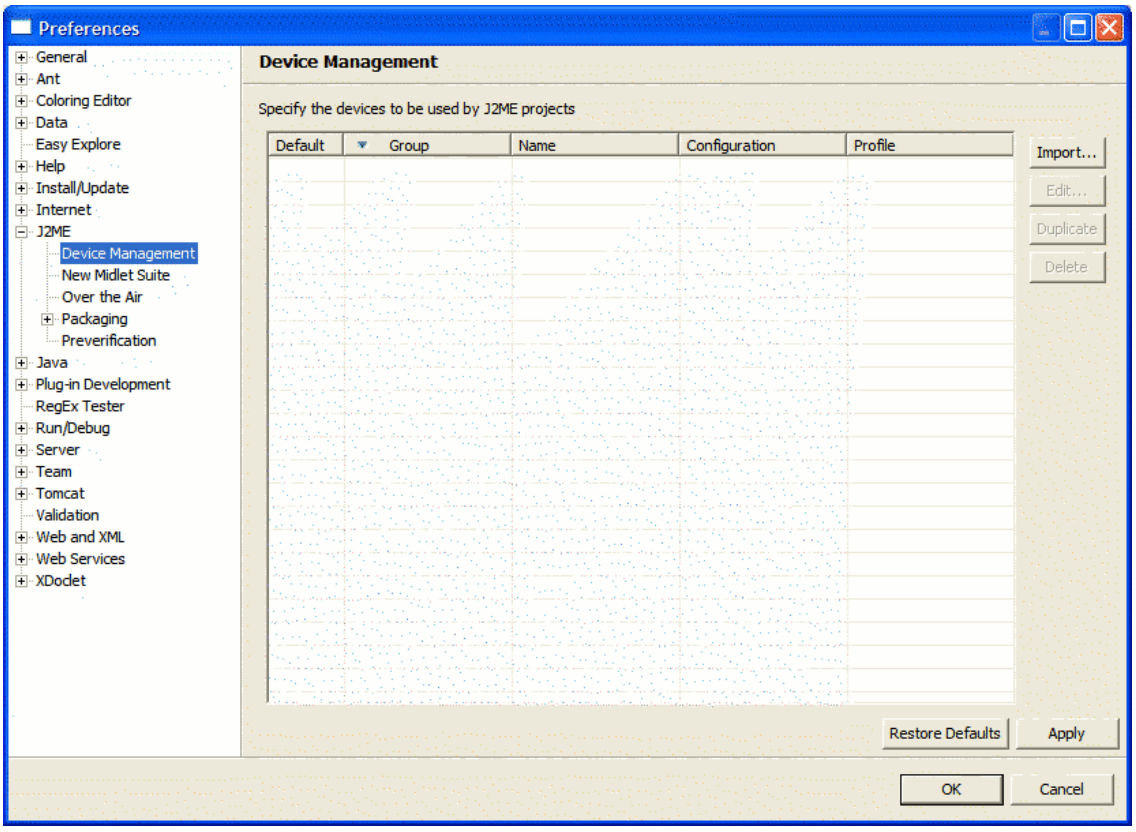
Eclipse 重启动之后，安装流程结束。
绝大多数情况下，上面的安装流程是可以正确更新旧版 EclipseME 的。
在极少数情况下，在 Eclipse 重启之后，EclipseME 可能仍未能正确“注册”到 Eclipse 中。在这种情况下，只要关闭 Eclipse，然后重启动 Eclipse 的时候增加一个 -clean 参数就可以了。 使用-clean 参数可以强制 Eclipse 重新扫描并更新它的所有插件信息。
如果正确安装了插件，那么在窗口/首选项对话框中会多出来一个 J2ME 选项分支。



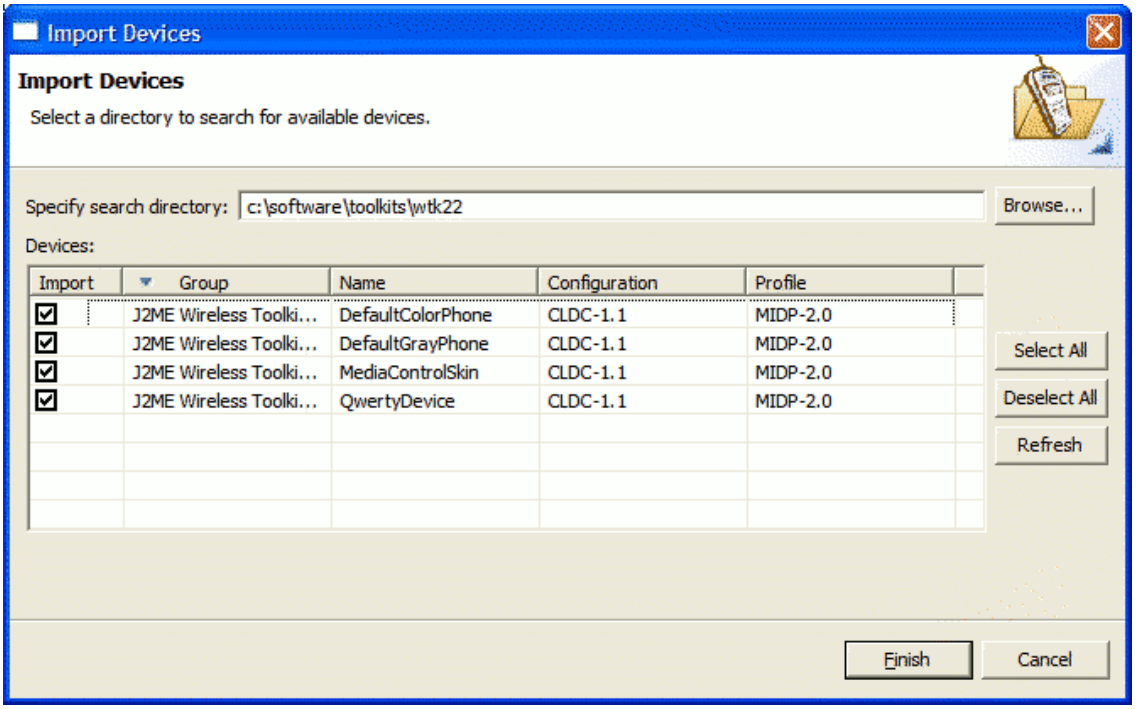
下面的步骤完成了上面的这些步骤之后，你可以开始配置 EclipseME 和 Eclipse 了。

3.3. 配置 EclipseME 和 Eclipse

1. 导入设备定义
你必须至少配置一种设备定义之后才能开始使用 EclipseME。请按下列步骤来配置设备定义，从 Eclipse 的窗口菜单中选择首选项。打开左边面板的 J2ME 选项分支，点击设备管理(Device Management)。



按导入...(Import)按钮。在接下来的对话框中，选择一个包含无线工具包的根目录，EclipseME 将从中查找已知设备定义。

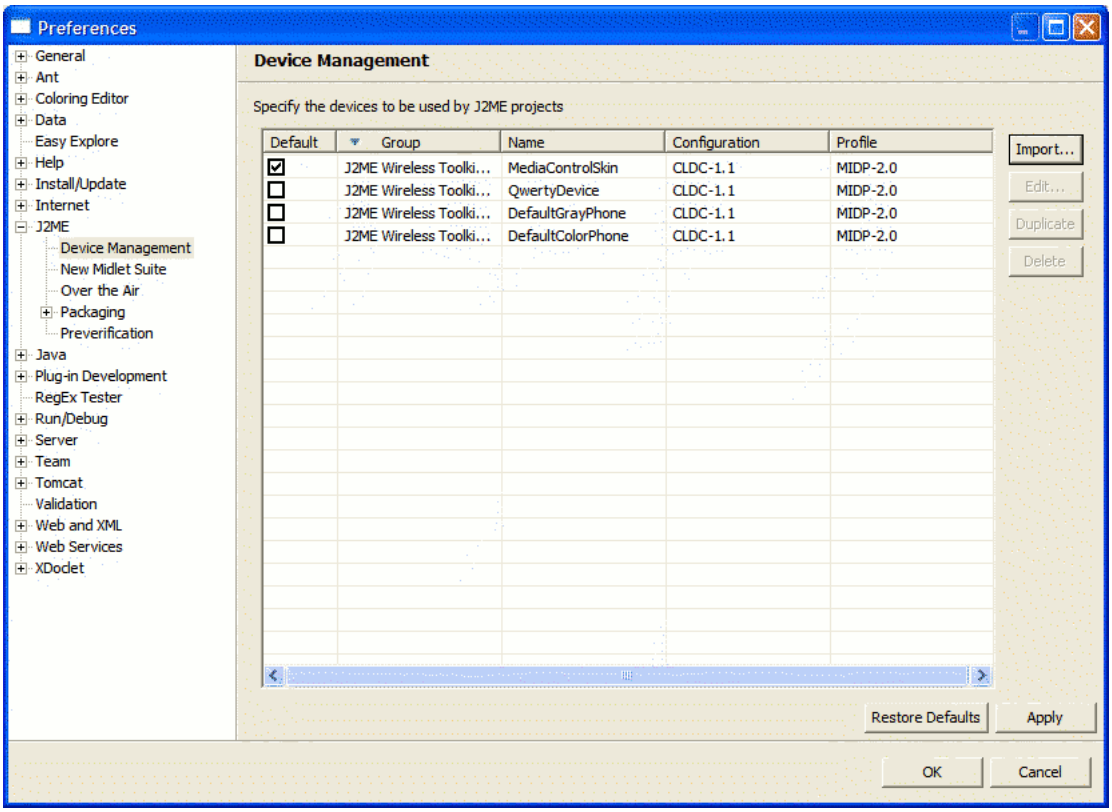


当你离开查找目录文本框，或按下了刷新(Refresh)按钮， EclipseME 就会开始从指定的目录和其子目录中搜索设备定义。 在 EclipseME 的 1.5.0 版本中，你不再需要“精确的”选择目录才能正确导入设备， 只要它们位于你指定的目录中，EclipseME 就应该能定位到它们。 找到一个设备定义，EclipseME 就会在列表中显示出一个。如果你希望停止搜索， 只要按下取消(Cancel)按钮就可以。

搜索结束以后，勾选你想导入的那些设备定义。只有导入后的设备才能被用于定义项目以及启动程序。 最后，选择完成(Finish)结束导入流程。

如果 EclipseME 没找到你希望导入的设备定义，那可能是 EclipseME 还不支持它。 这种情况下，请向我们提交一个 RFE(新特性需求)来要求增加对这种 WTK 的支持，别不好意思。 具体怎么做请参见这里。

当你成功的增加了设备定义，你就会在设备管理首选项中看到这些导入的设备。



点确定来保存设备定义。

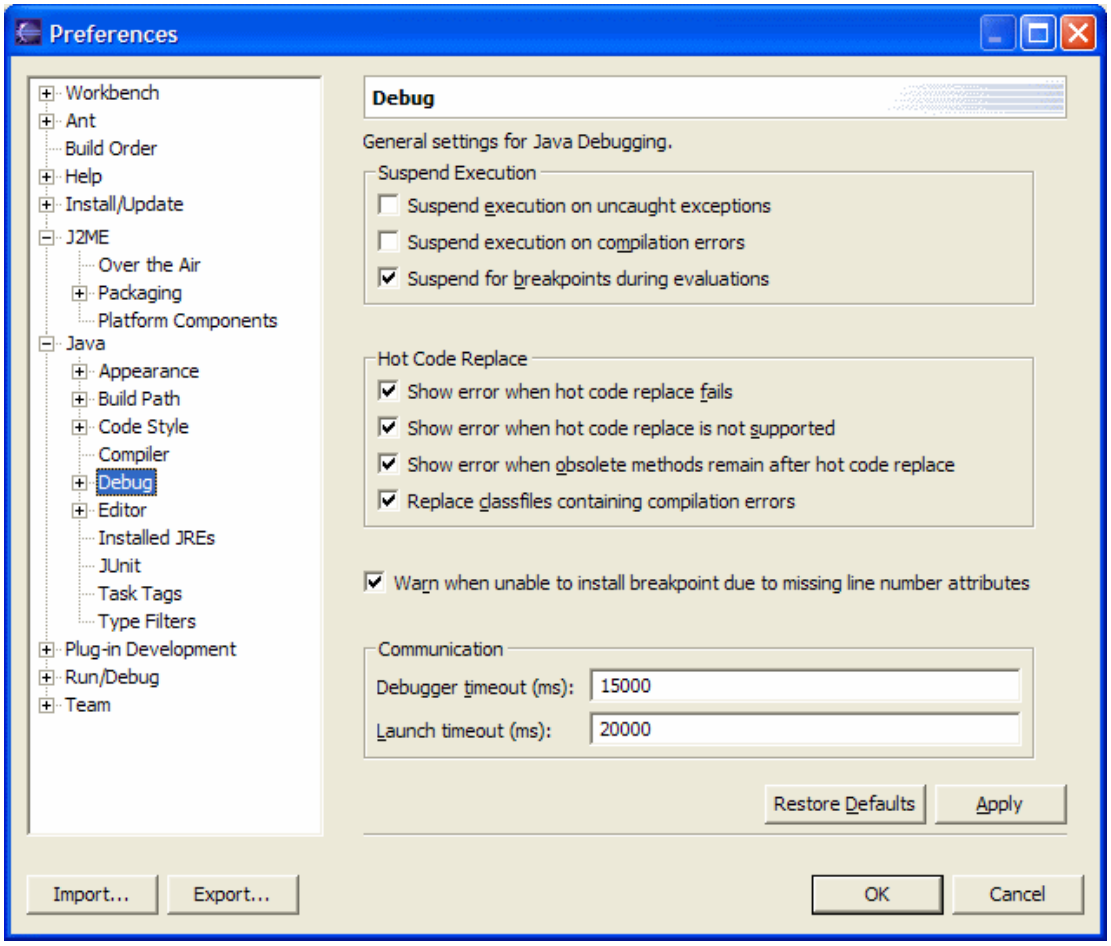
2. 改变 Eclipse 的调试设定

因为无线工具包有一些奇怪的设计（尤其是 Sun 的），如果你想使用 Eclipse 来调试 MIDlet，那你必须改变一些默认的调试设定。像这样：

选择 Eclipse 的窗口菜单中的首选项。

打开左边面板中的 Java 选项分支 并点击调试项。

确保接近对话框顶端的两项：发生未捕获的异常时暂挂执行 和在发生编译错误时暂挂执行都没有被勾选。 把接近对话框底部的：调试器超时的值增大到最少 15000 毫秒。 设定后的结果应该看上去类似这样：



如果你不做这些改变，在运行 MIDlet 的时候就会发生错误。

3. 配置 ProGuard 混淆器（可选）

如果你想用 ProGuard 来产生混淆包，那你就需要把它配置到插件中。这样做：

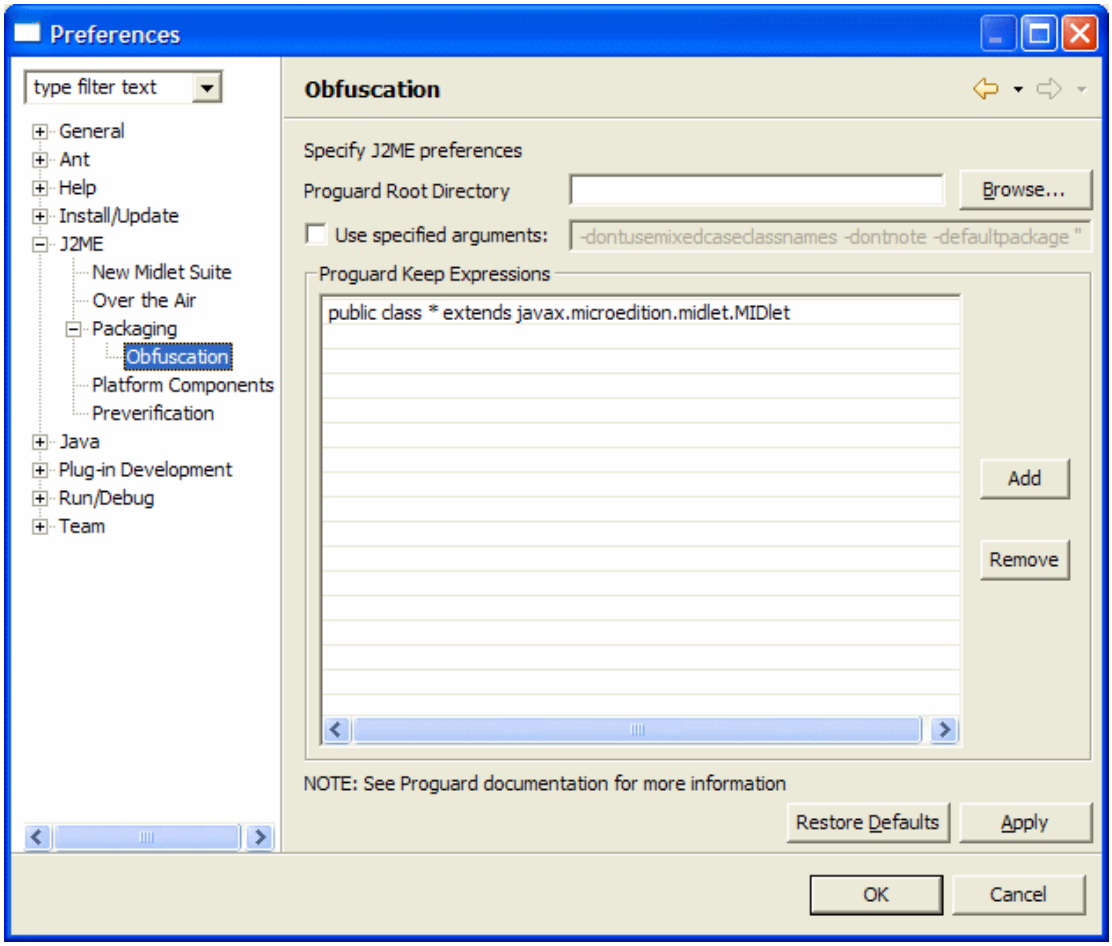
选择 Eclipse 的窗口菜单中的首选项。

打开左边面板的 J2ME 选项分支， 选择打包(Packaging)子项， 然后点击混淆(Obfuscation)。

配置 ProGuard 的根目录（在对话框顶部）。

在下面按照你的需要配置ProGuard选项。更多关于ProGuard的信息，请参见 ProGuard的SourceForge站点。

Microsoft Windows 用户请注意： 默认情况下，ProGuard 假定你使用的的操作系统能够区分两个只是大小写不同的文件名的 （比如，A.java 和 a.java 被认为是两个不同的文件）。 显然 Microsoft Windows 不是这样的操作系统（Windows 是对文件名是大小写不敏感的）。 因此 Windows 用户必须为 ProGurad 指定-dontusemixedcaseclassnames 选项。 如果你不这么做并且你的项目中有超过 26 个类的话，那么 ProGuard 就会默认混用大小写文件名， 而导致 class 文件相互覆盖。安全起见，从 0.9.0 版本开始，EclipseME 默认为 ProGuard 设置 -dontusemixedcaseclassnames 选项。项目中有很多类的 UNIX 用户 可以删除这个选项，这样最终产生的 JAR 文件的大小可以进一步缩小。 设定后的结果应该看上去类似这样：

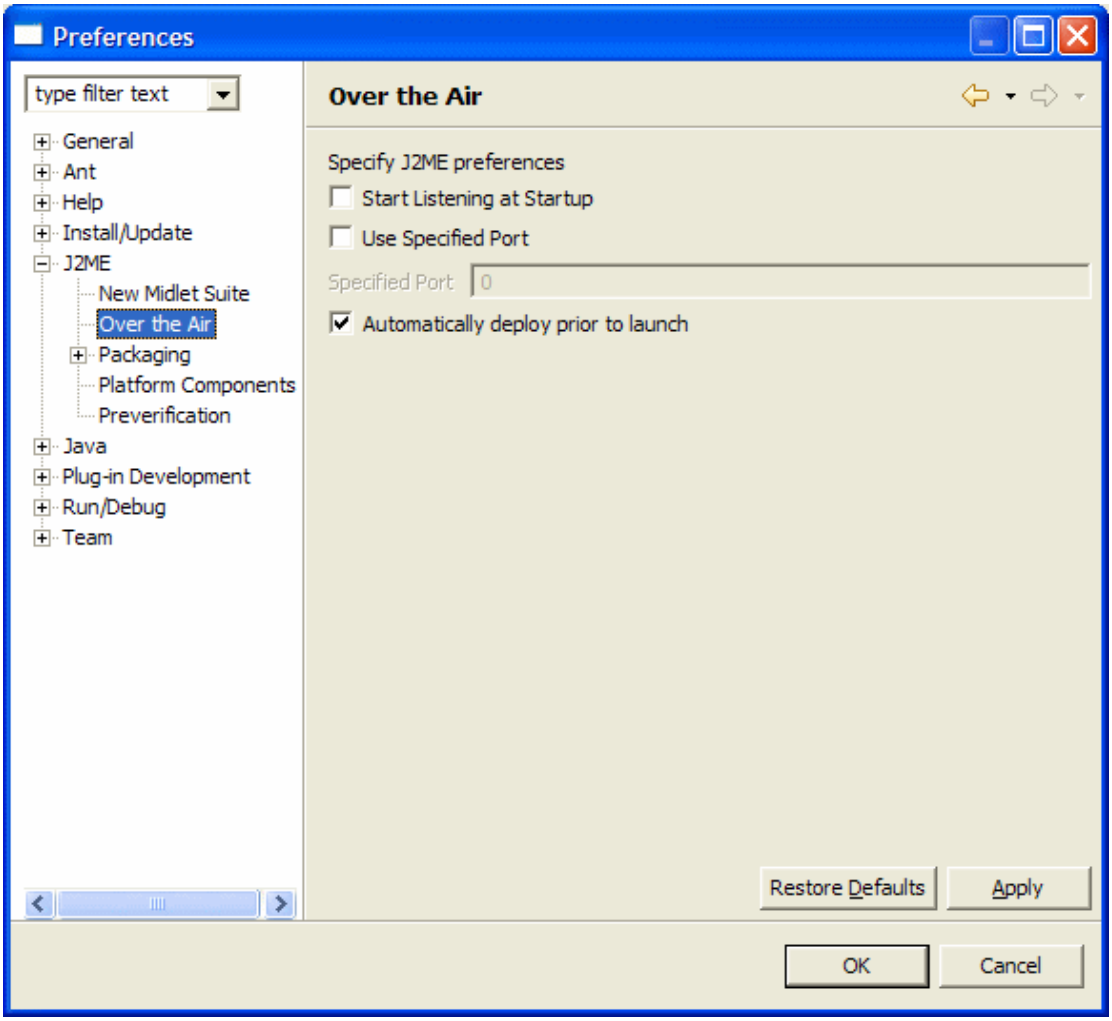


4. 配置 EclipseME 的无线下载(OTA)选项（可选）

如果你想使用 EclipseME 在无线下载(OTA)模式下调试 MIDlet，那么你可能需要调整 OTA 首选项。这样做：

选择 Eclipse 的窗口菜单中的首选项。

打开左边面板中的 J2ME 选项分支 并点击 Over The Air 项。 默认的设置应该是这样：



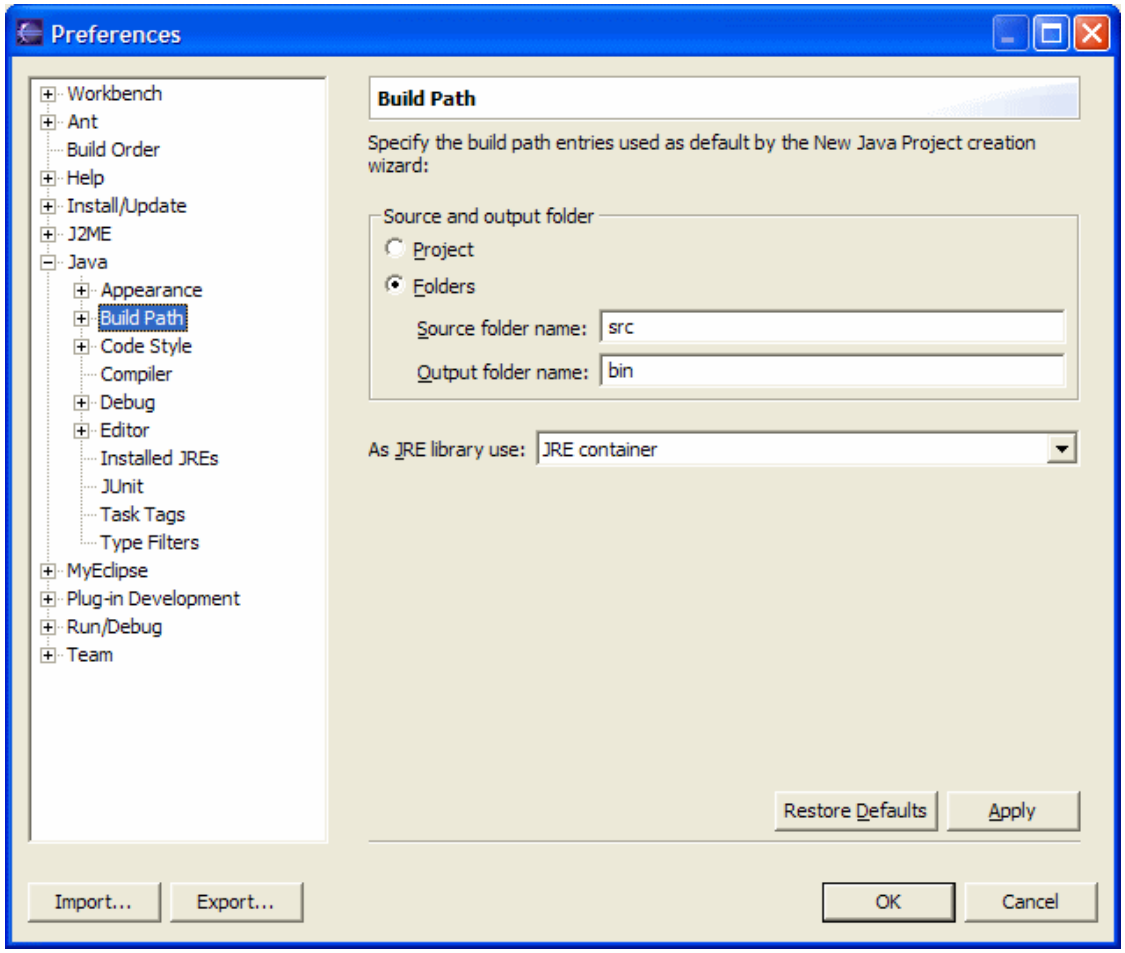
到这里，你已经准备就绪，可以开始创建你的 MIDlet 套件项目了。

4. 最佳实践

本文档提供了一些关于使用 EclipseME 的“最佳实践”的讨论。 虽然这些并不是强制性的，但是我们认为，按照下面的这些指南来进行开发将使你感到更轻松。

- 使用单独的源文件夹和输出文件夹

虽然并非强制，但我们非常强烈的建议你把 Eclipse 配置成在新项目中使用单独的源文件夹和输出文件夹。 很遗憾的，这不是 Eclipse 的默认设置。 要改变这个设置，请在 Eclipse 的“窗口”菜单中选择“首选项”， 然后在对话框的左边面板展开“Java”分支并点击“构建路径”。在右边的面板选中“文件夹”单选按钮。 然后对话框看上去应该是这样的：



如果你没有把 Eclipse 配置成使用单独的源文件夹和输出文件夹，EclipseME 的一些功能，比如资源文件夹，就无法使用了。

如果你希望转换一个未使用单独源和输出文件夹的项目，可以按照下面的步骤来操作：

1. 在你项目的根目录下创建一个 src 文件夹。
2. 选择“项目”菜单的“属性”菜单项， 然后在左边面板选择“Java 构建路径”项。
3. 点击“添加文件夹”按钮， 把你先前创建的 src 文件夹左面的复选框打上对勾，然后按“确定”。
4. Eclipse 会提示你想要除去作为源文件夹的项目并将构建输出文件夹更新为“[项目]/bin”吗？ 按确定，然后按确定按钮关闭项目属性对话框。
5. 这时 Eclipse 会提示你输出文件夹已更改。想要除去旧位置“/[项目]”中已生成的所有

资源吗？ 回答“是”。

- Java 兼容等级

Java 兼容等级包括两方面。第一是指编译器编译时可以接受的源文件语法。比如，Java 1.4 引入了 assert 关键字。如果在源代码兼容级别低于 1.4 的情况下编译一个包含 assert 语句的源文件，那么 assert 语句就会被视为语法错误。

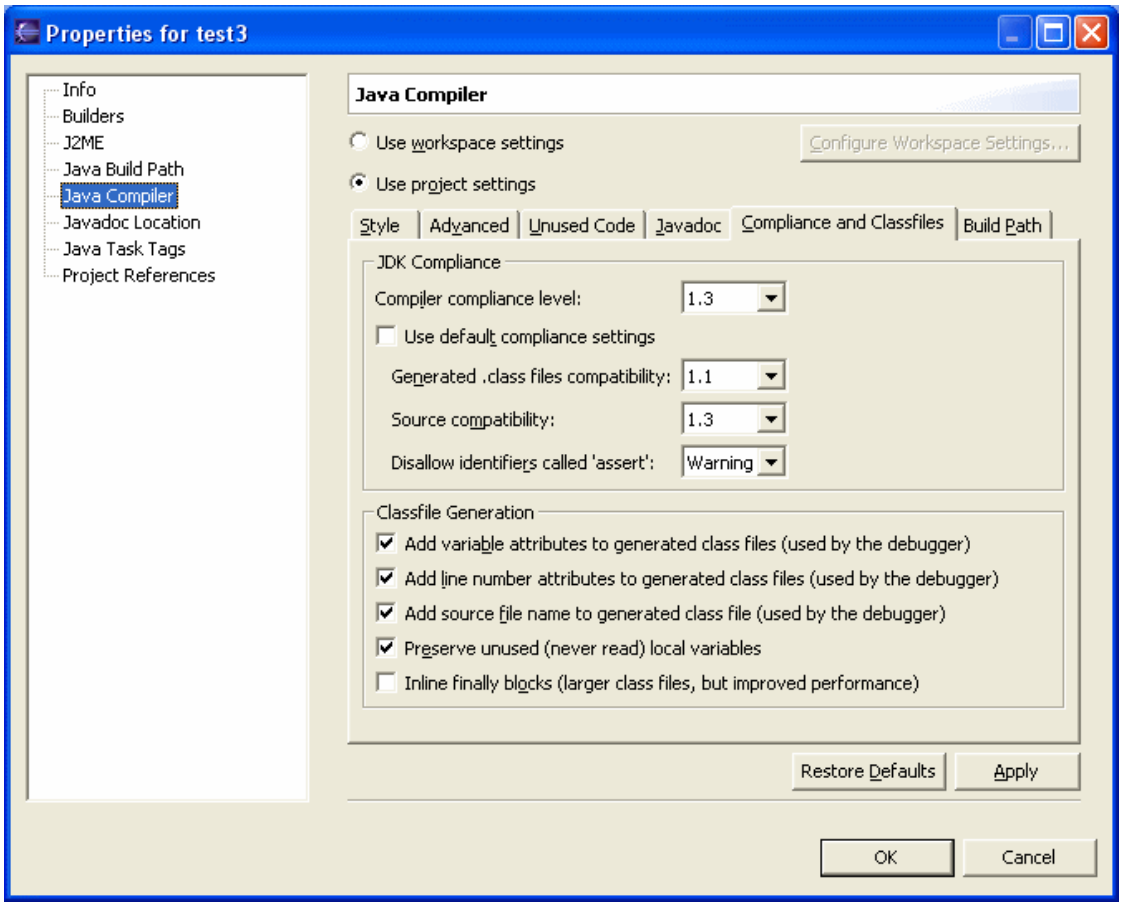
兼容性级别的第二方面是指 Java 编译器产生的 class 文件的内部格式。 随着 Java 语言的发展，class 文件的内部格式上已经有了一些微小的变化。 新的 Java 虚拟机实现可以接受旧的 class 文件格式，但旧的 Java 虚拟机实现则不能接受新的格式。

很多 J2ME 设备不能接受比 1.1 更新的 class 格式。 不幸的是，很多针对这些设备的模拟器却可以接受新的格式，而没有任何警告。 这会导致很大的混乱，因为 MIDlet 在模拟器上运行良好，但会被实际设备拒绝。

因此，为了减少麻烦，便于安装，我们推荐你总是把你的项目设置为按照 Java 1.1 格式输出。 就我们所知，在那些接受新格式的设备上这也不会导致任何兼容性问题。

从 EclipseME 0.9.0 版本开始，只要在"New Midlet Suite"首选项中选中了"Force Java 1.1 compliance"（强制兼容到 Java 1.1），那么新建的 MIDlet 套件项目就会自动按照上述方式设置。 你可以按照下面的步骤来调整一个现存项目的兼容性级别：

1. 选择“项目”中的“属性”菜单项来打开项目的属性对话框。
2. 在左边面板点击“Java 编译器”分支， 然后右面就是兼容性和 Class 文件面板。
3. 按照下图调整设置，按“确定”。



- 资源目录

相对于前两项，使用资源目录是一个很个人的选择。 有些开发者坚持认为非 Java 文件不应该出现在源文件所在的位置，但是也有些人认为把 Java 源文件和资源文件混放在一起没

什么问题。 Eclipse 本身对这点没有明确的约定。事实上，Eclipse 是把资源目录视为源文件目录同等对待的。

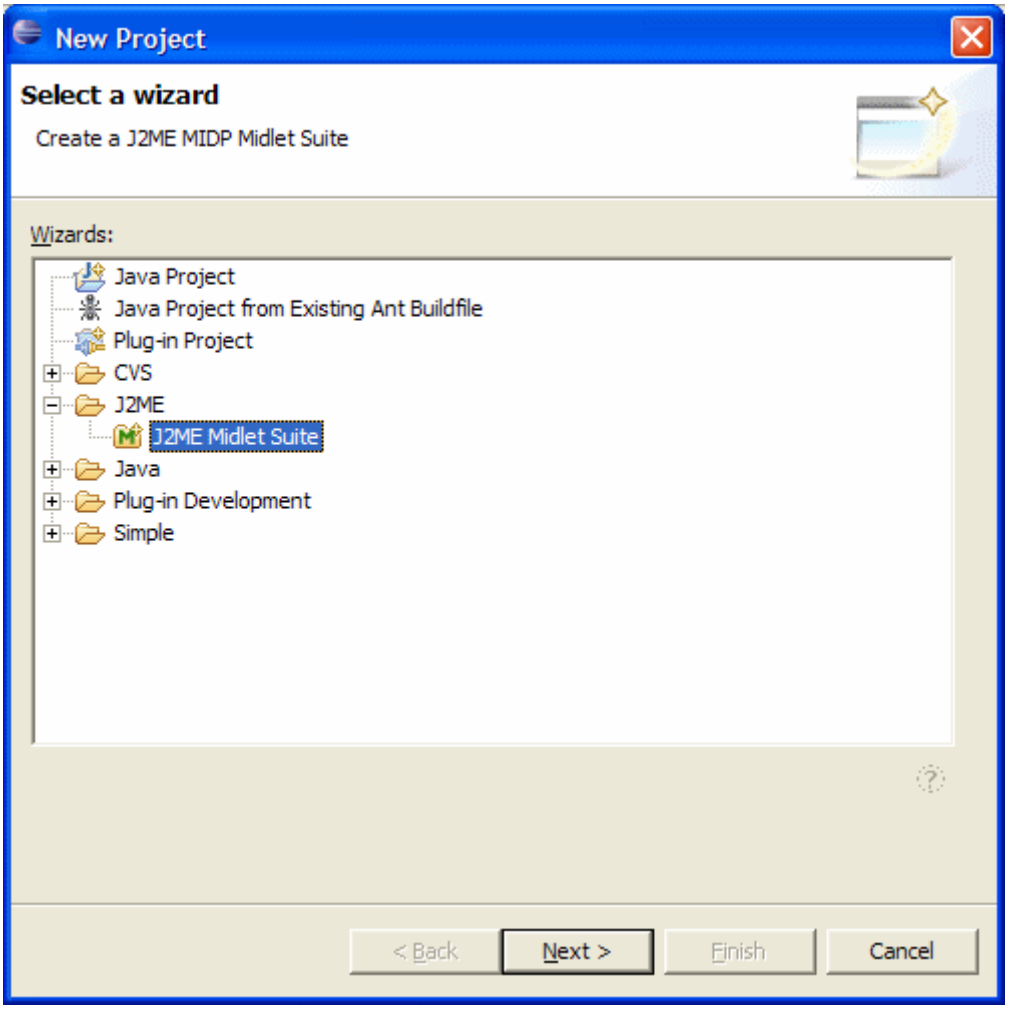
如果你希望在创建项目时自动生成一个资源目录，只要确认在 "New Midlet Suite" 首选项 面板中勾选了对应的复选框并为资源目录取一个你喜欢的名字就可以。

如果你希望为现存项目增加一个资源目录，只要在其工作空间中创建文件夹， 并在项目属性的“Java 构建路径”中把它添加到 “源代码”面板里就可以了。

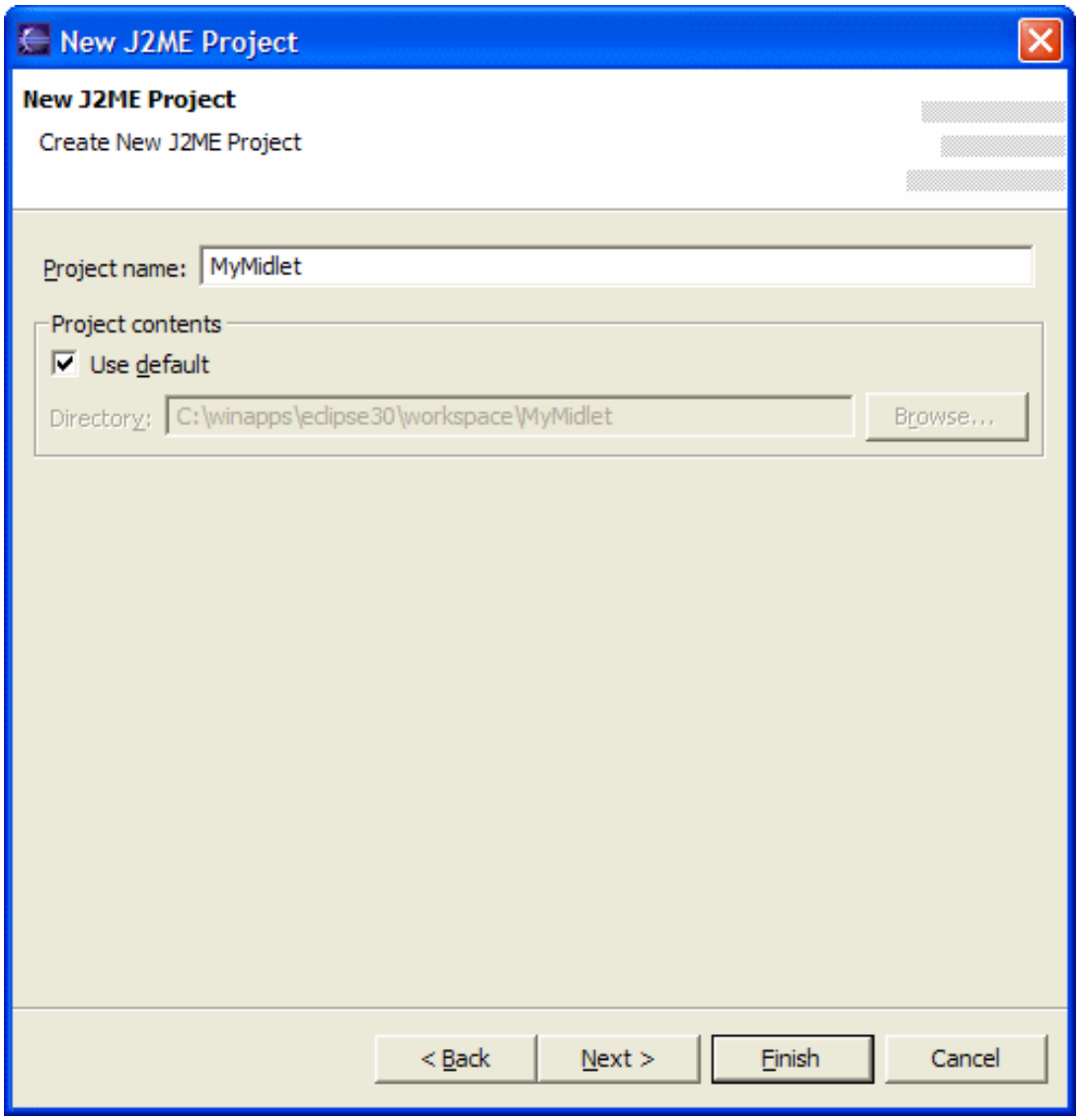
5. 创建新的 J2ME MIDlet 项目

5.1. 从头开始创建一个新项目

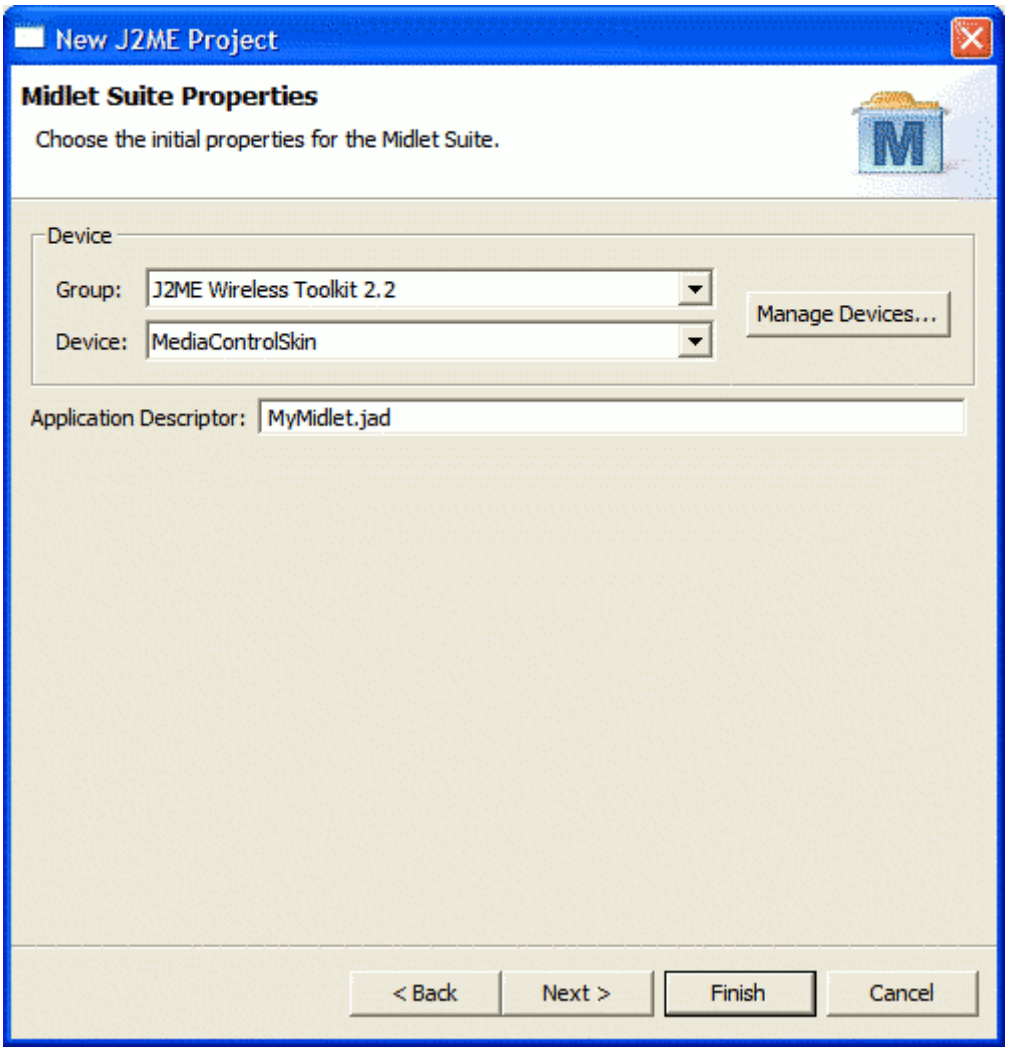
使用新建项目向导，选择 J2ME 中的 J2ME Midlet Suite 项来创建一个 EclipseME 项目。



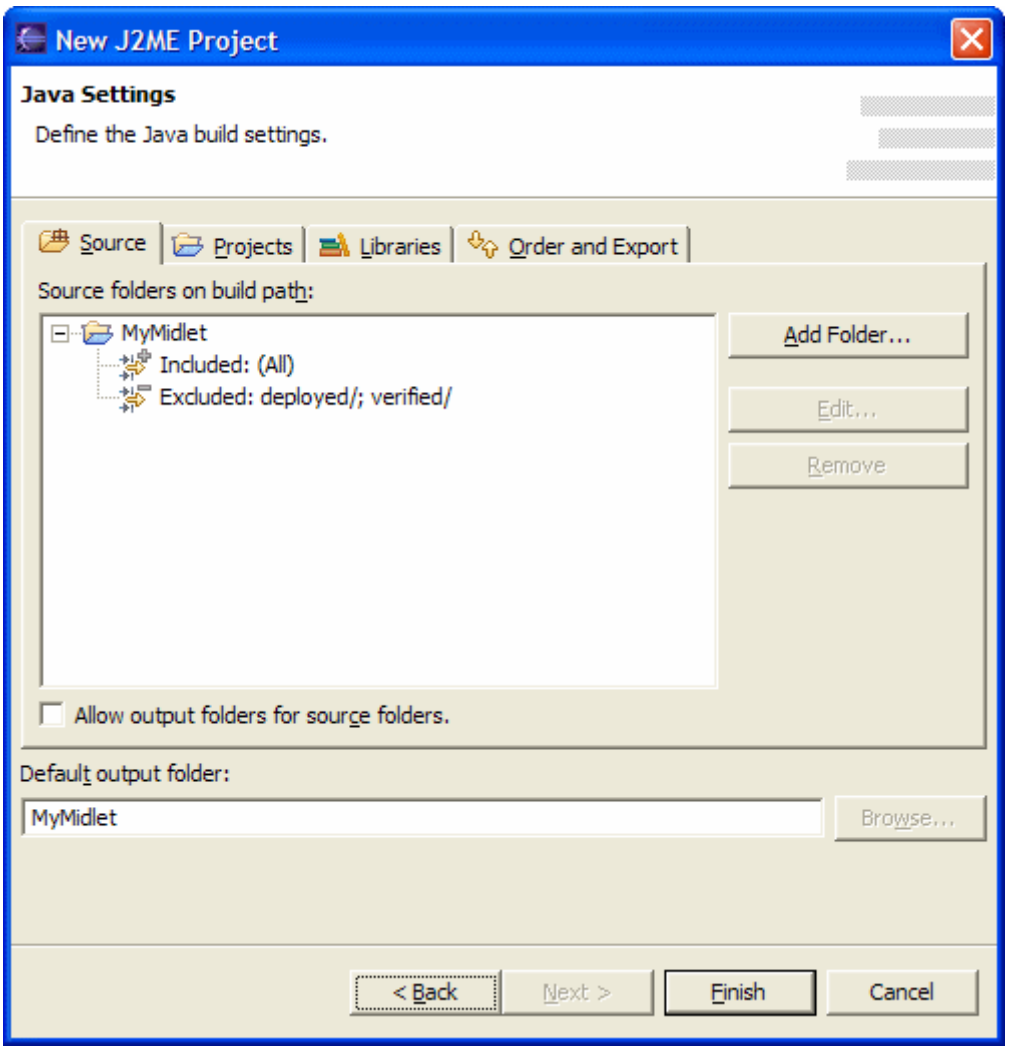
向导的第一个界面和标准 Eclipse 项目的一致。 在这里，你要给出项目的名字，以及项目的位置。



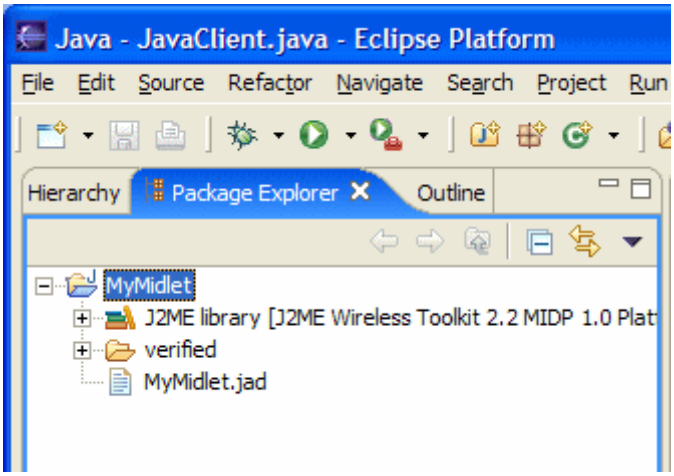
在第二个界面里，你要从可用的设备定义中选择一个，用于编译、运行和调试项目。 若有必要，以后你可以在项目属性中改变这个设定。 另外，在这个界面中，你也可以改变项目的 JAD 文件的默认位置。 在这里指定的 JAD 文件必须相对于项目的根目录来创建。 以后你可以在包资源管理器中使用重构（重命名或移动）来修改 JAD 文件位置。



最后的界面让你可以调整源文件设置，相关的项目和类库，等等。这是 Eclipse 的标准设置界面。 如果你的项目需要外部的类库（比如 kXML 等）， 只要把相应的 JAR 文件增加到“库”(Libraries)这个面板的列表中就可以了。 EclipseME 会自动把这些 JAR 文件和你的 class 文件一起进行管理。

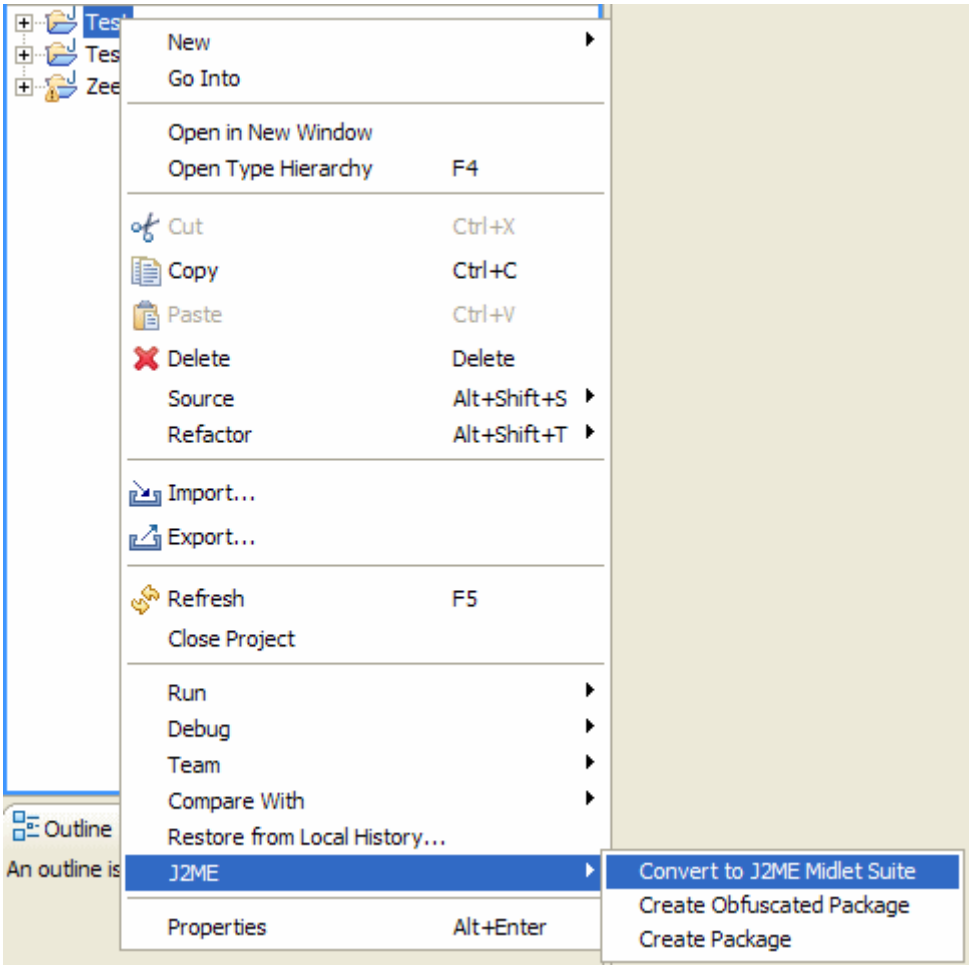


你完成了上面的这些步骤之后，在你的工作空间里面你就会看到类似下面的这样一个项目：



5.2. 把现成项目转换为 EclipseME 项目

如果你有一个现成的 Java 项目，并希望把它转换成 EclipseME 项目，只要从项目的弹出菜单中选择转换成 J2ME MIDlet 套件(Convert to J2ME Midlet Suite)就可以了。



当你这么做的时候，EclipseME 插件会自动提示你选择设备定义，并对你的项目进行一些相应的修改。

如果把一个 J2SE 项目转换成 EclipseME 项目，那么需要从构建路径中删除 J2SE 的类库。EclipseME 将自动删除可以识别的类库。 如果 EclipseME 没能成功删除 J2SE 类库，那么你需要自己手动删除它们。

EclipseME 同时也会自动向你的项目中加入适当的 J2ME 类库。 特定的类库是根据项目选择的设备定义来加入的。如果你在把项目转换到 EclipseME 项目之前 就使用 Eclipse 来进行 J2ME 开发，那么在你的项目构建路径中可能仍然会保留一些 J2ME 类库。 如果这样，你必须手动删除它们。否则可能发生问题，因为 EclipseME 可能会认为它们是 项目依赖的第三方类库，而错把它们打包到你的 JAR 文件中去。

5.3. Java 应用描述文件(JAD)编辑器

- 概述

创建了新的 MIDlet 套件项目之后，你可能会需要更改套件的属性。 要编辑 Java 应用描述文件(JAD)来修改 MIDlet 套件的属性。

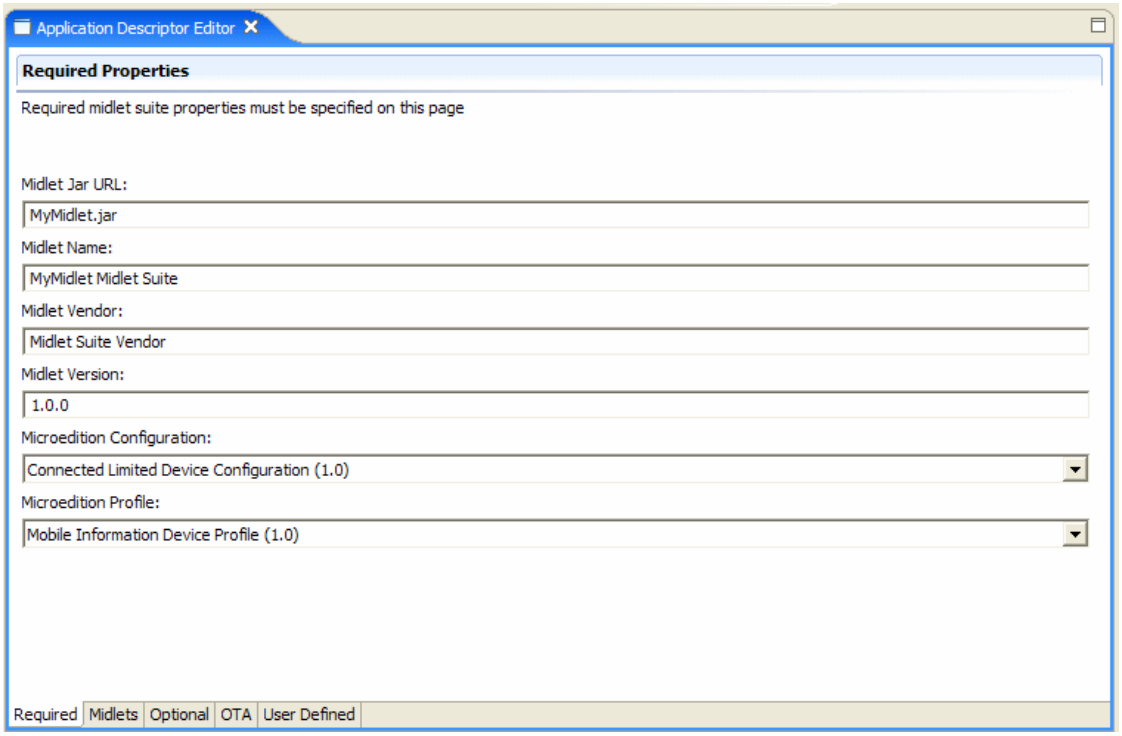
J2ME 发布包由一个包含软件的 JAR 文件，和一个用来向 J2ME 容器描述源文件内容相关信息的 JAD 文件组成。

EclipseME 自带了一个 JAD 编辑器，帮你处理 JAD 文件的格式等细节问题。 使用这个编辑器，你可以填入所有必须的项目，使设备能够正确的支持你的 MIDlet。 方便起见，组成 JAD 文件

的项目分布在编辑器的几个不同面板中， 编辑器窗口底部的一排就是这些面板的标签。

- 必选属性(Required)面板

JAD 编辑器的第一个标签对应的是必选属性(Required)面板。



此面板上的项目是：

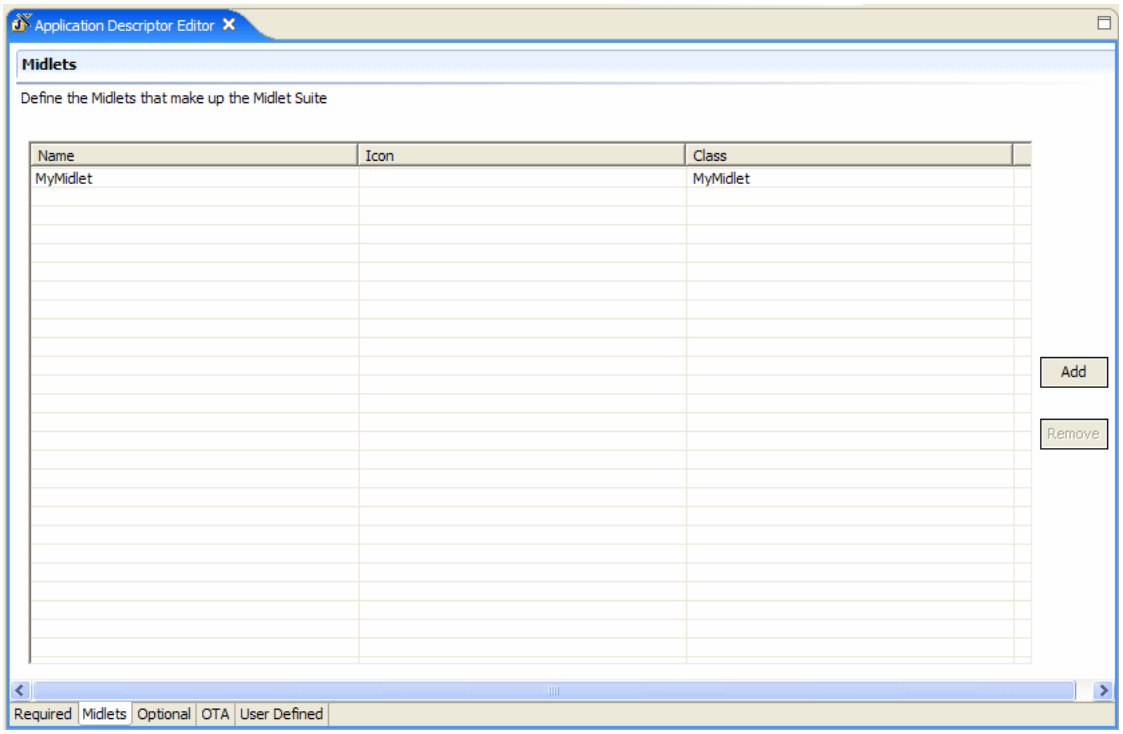
项目	内容
Midlet Jar URL	JAR 文件的 URL 地址。
Midlet Name	此 MIDlet 套件的名字，也就是用户看到的本软件的名字。 套件中的每一个 MIDlet 也可以有自己的名字。请参见下面的 Midlets 面板。
Midlet Vendor	本 MIDlet 套件的开发商/提供商。
Midlet Version	本 MIDlet 套件的版本号。按照 Java 产品版本号命名规范(Java Product Versioning Specification) 中的约定，格式是<主版本号>.<副版本号>.<微(micro)版本号>。 J2ME 容器可以使用本信息来进行软件的安装和升级，以及为用户提供信息提示。
Microedition Configuration	本 MIDlet 套件运行所必需的 J2ME Configuration (CLDC)的版本。在这里的下拉列表中可选 择的内容决定于项目所选的无线工具包可支持的 CLDC 版本。
Microedition Profile	本 MIDlet 套件运行所必需的 J2ME Profile (MIDP)的版本。和 Microedition Configuration 一样， 在这里的下拉列表中可选择的内容决定于项目所选的无线工具包可支持的 MIDP 版本。

注意 EclipseME 插件会自动处理 J2ME 规范中要求的必选项 MIDLet-Jar-Size， 因此你无需输入此信息。

- Midlets 面板

编辑器窗口的第二个标签对应的是 Midlets 面板。 在这个面板上，你必须保证 MIDlet 套件

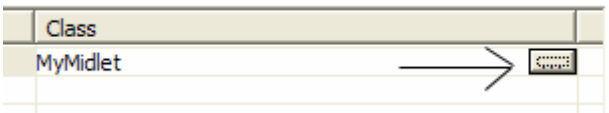
中的每一个 MIDlet 都要有一个对应项。 如果你在创建 MIDlet 类的时候选中了 添加到应用描述文件(Add to Application Descriptor)复选框， 那么其对应项会被自动添加。否则，你就必须使用添加(Add) 按钮来为你的 MIDlet 添加其对应项。
在 EclipseME 的 0.5.0 版本以前，按添加(Add)按钮会弹出一个对话框， 供你填入相关信息。从 0.5.0 版本以后，这个面板使用起来更像一个电子表格软件了。 按添加(Add)按钮会创建一个新行，而行里面的项目可以直接编辑。



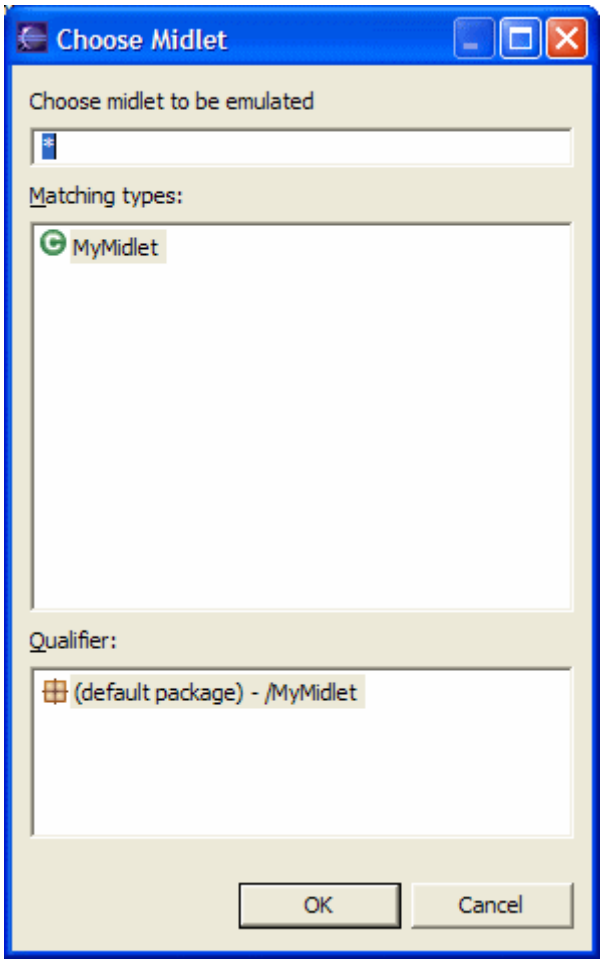
此面板上的列分别是：

项目	内容
名称	MIDlet 的名称。如果你的套件中包含多个 MIDlet，那么当用户打开套件时，J2ME 设备通常会 提示用户选择一个 MIDlet 来运行。此项目提供了显示给用户的名字。
图标	作为此 MIDlet 的图标的 PNG 文件在 JAR 文件中的路径。
类名	MIDlet 类名。这是一个你从 javax.microedition.midlet.MIDlet 派生的子类。

编辑名称和图标路径，只要简单的点击单元格并输入新的值。如果要改变类名，则点击单元格， 这时一个按钮会出现。

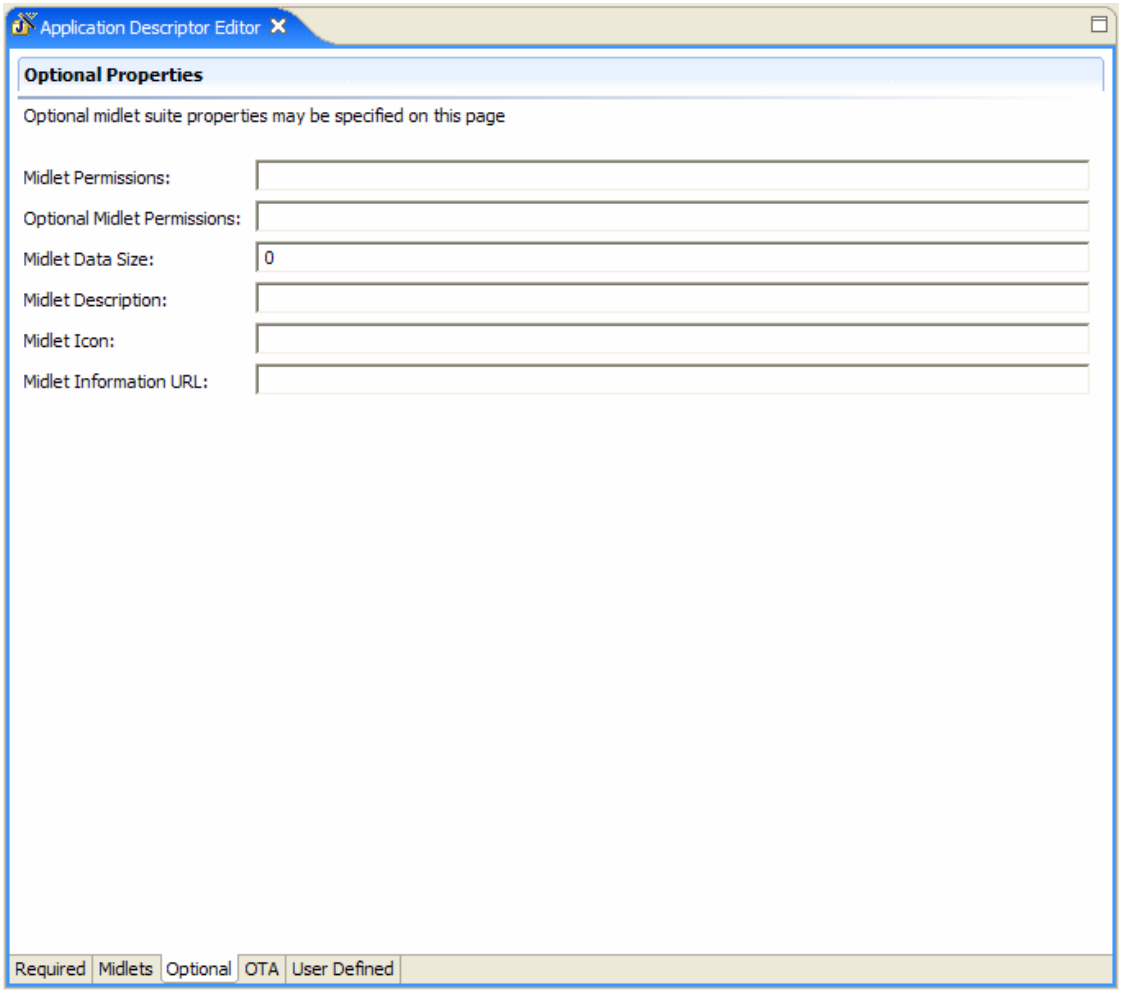


按下这个按钮会弹出一个对话框，允许你选择正确的 midlet 类。



- 可选属性(Optional)面板

编辑器窗口的第三个标签对应的是可选属性面板。 在这个面板上你可以编辑那些在 J2ME 规范中定义的但非必需的属性。

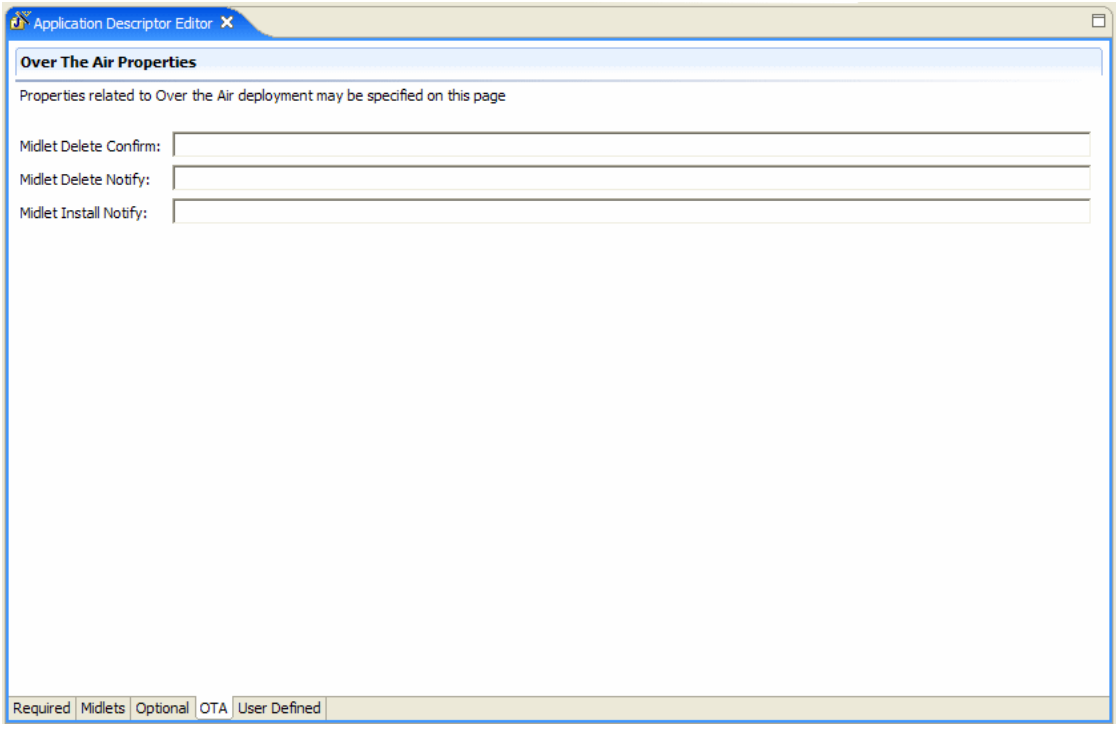


此面板上的项目是：

项目	内容
Midlet Permissions	你的 MIDlet 正常运行所必需的权限。通常只有经过数字签名的 MIDlet 才需要提供权限属性。
Optional Midlet Permissions	你的 MIDlet 所希望获得的权限，即使没有这个权限 MIDlet 也可以运行。
Midlet Data Size	MIDlet 保存持久数据所需要的最小字节数。 根据自己的安全级别设定，设备可能提供更多的空间给 MIDlet。 默认值是 0。
Midlet Description	MIDlet 套件的描述。
Midlet Icon	作为 MIDlet 套件的图标的 PNG 文件在 JAR 文件中的路径。这是 Java 应用管理器用来标识 MIDlet 套件的图标。 这个图标是针对整个 MIDlet 套件的，请区别于在 Midlets 面板 中设定的针对单个 MIDlet 的图标。
Midlet Information URL	一个可以获得 MIDlet 套件进一步信息的 URL。

- 无线下载属性(Over the Air)面板

编辑器窗口的第四个标签对应的是无线下载属性(Over the Air)面板。 在这个面板上你可以编辑跟无线下载规范相关的属性。

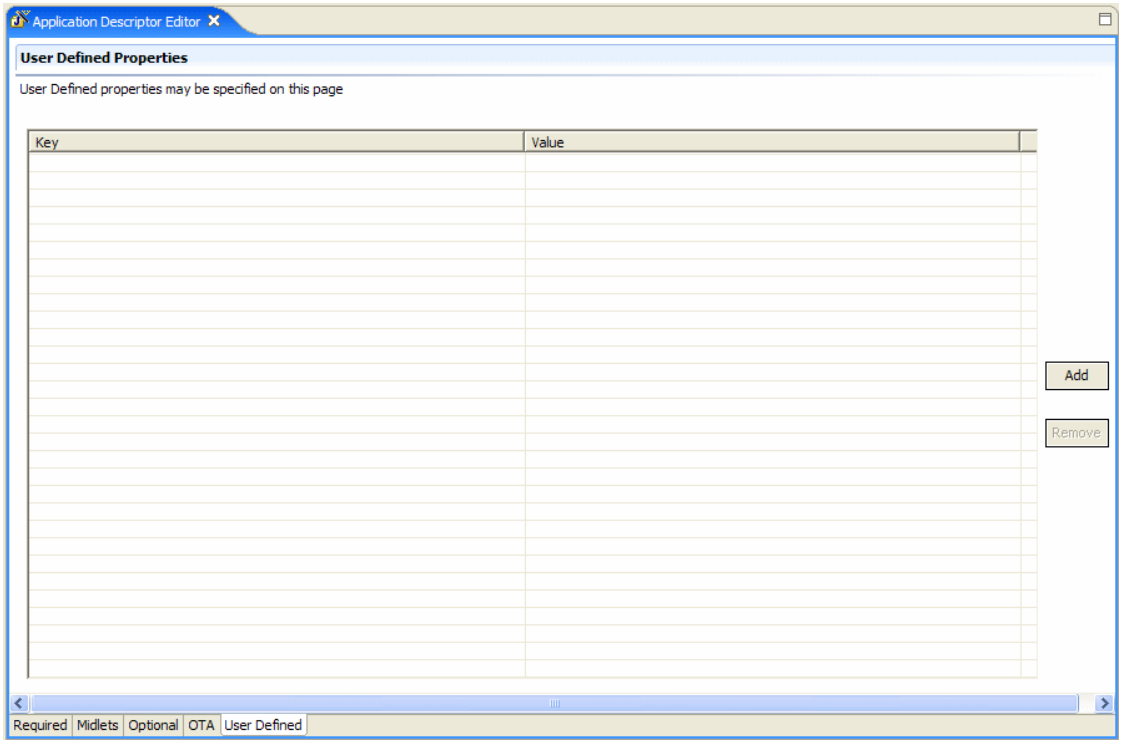


此面板上的项目是：

项目	内容
Midlet Delete Confirm	当用户要删除 MIDlet 套件时弹出的确认提示信息。
Midlet Delete Notify	删除通知 URL，用来向其发送一个 POST 请求来确认 MIDlet 套件删除成功。
Midlet Install Notify	安装通知 URL，用来向其发送一个 POST 请求来确认 MIDlet 套件安装成功。

- 用户自定义属性(User Defined)面板

编辑器窗口的第五个标签对应的是用户自定义属性(User Defined)面板。 在这个面板上你可以定义跟你的特定 MIDlet 相关的属性。

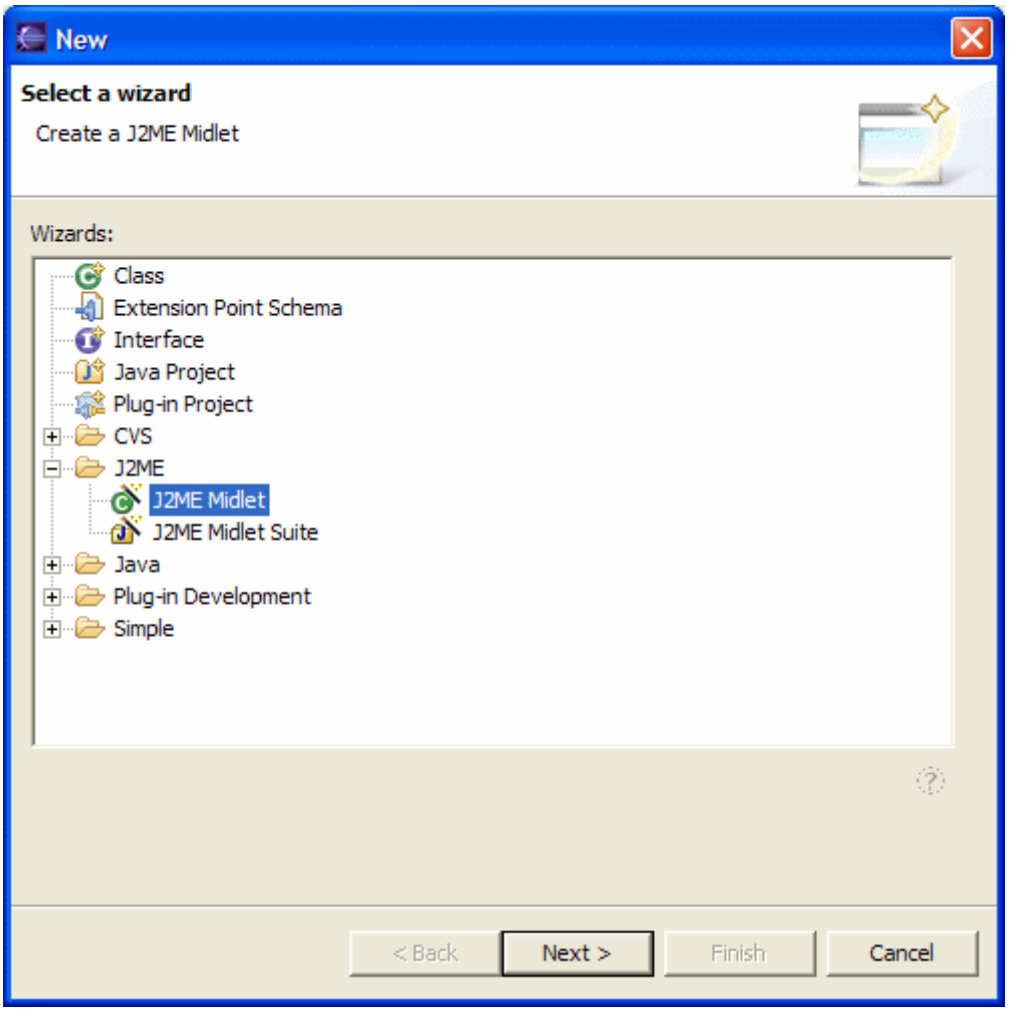


此面板上的列分别是：

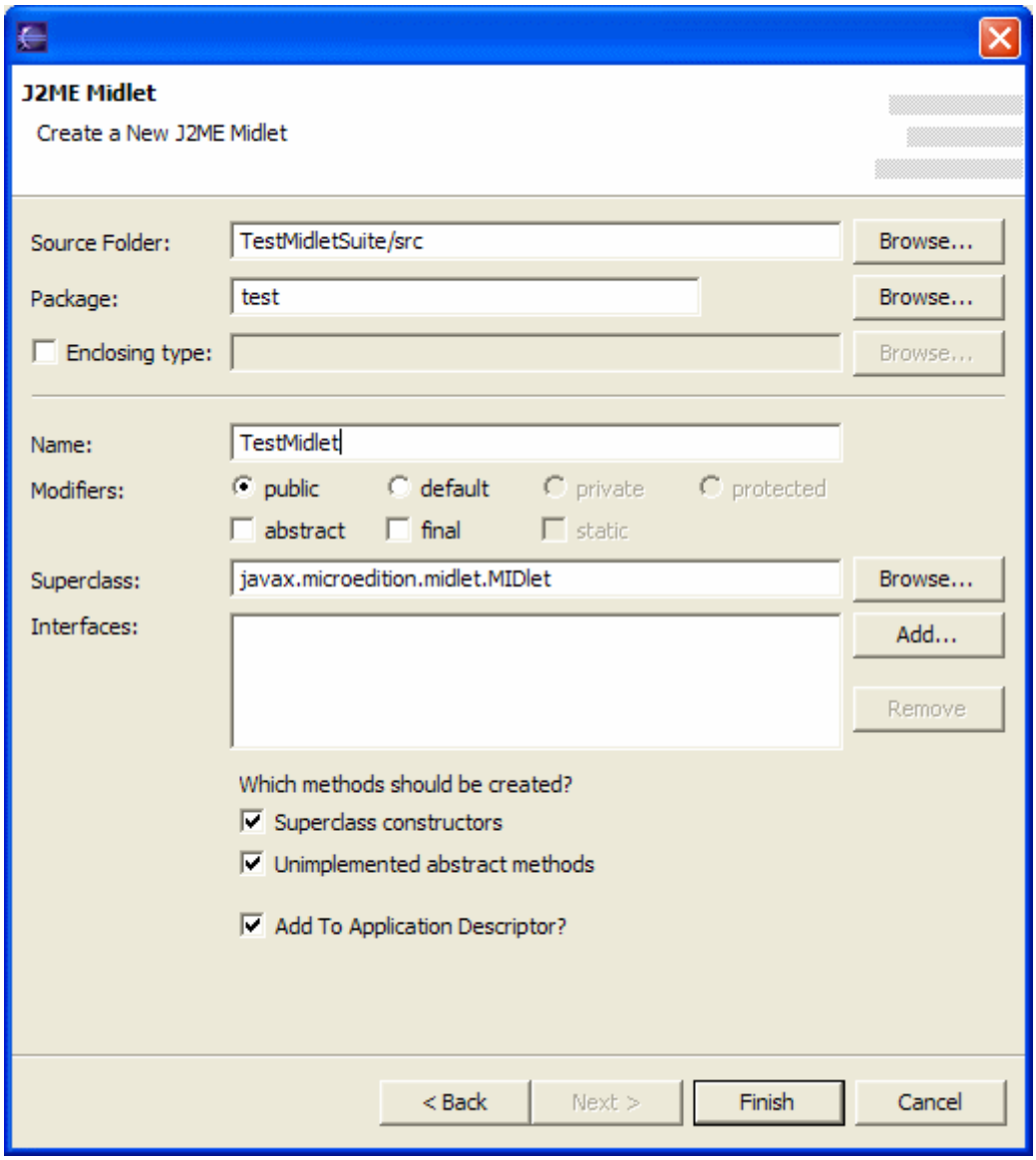
项目	内容
键(Key)	用来获取对应值的键字符串。
值(Value)	与键相对应的值。

6. 创建新的 MIDlet

创建一个新的 MIDlet，只要简单的创建一个从 javax.microedition.midlet.MIDlet 派生的新类就可以了。 如果你点击新建按钮并选择其它， 那么可以轻松的使用下面的选项来创建 MIDlet：



或者，你也可以使用标准的新建类，并手动指定 javax.microedition.midlet.MIDlet 作为基类。 在接下来的面板中，你可以设定新 MIDlet 子类的详细内容：



在这个对话框中，为新建的 midlet 类选择/填入适当的信息。 除了添加到应用描述文件(Add to Application Descriptor)这个复选框之外， 其它项目都是标准的类创建参数。

只有你使用前面描述过的创建新的 J2ME Midlet(New J2ME Midlet) 这个方法来创建类的情况下， 添加到应用描述文件(Add to Application Descriptor)复选框 才会出现。 如果选中了这个参数，那么 EclipseME 会使用默认信息自动把新创建的 midlet 加入到项目的应用描述文件(JAD)中。 以后可以使用 JAD 编辑器来修正 midlet 的信息。

如果你没有选中添加到应用描述文件(Add to Application Descriptor)复选框， 或者你是手动创建的 MIDlet 类，那么你必须使用 JAD 编辑器 来输入 MIDlet 的详细信息使得它能够被正确的注册为 MIDlet 套件的一部分。 否则 J2ME 设备无法访问此 MIDlet。

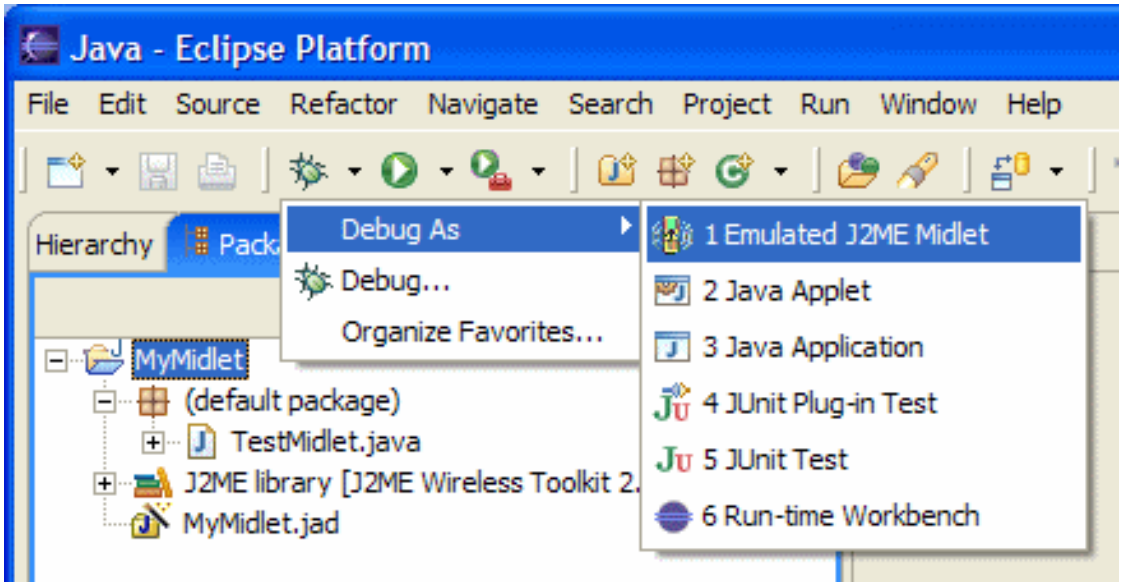
7. 运行/调试一个 MIDlet

7.1. 开始之前

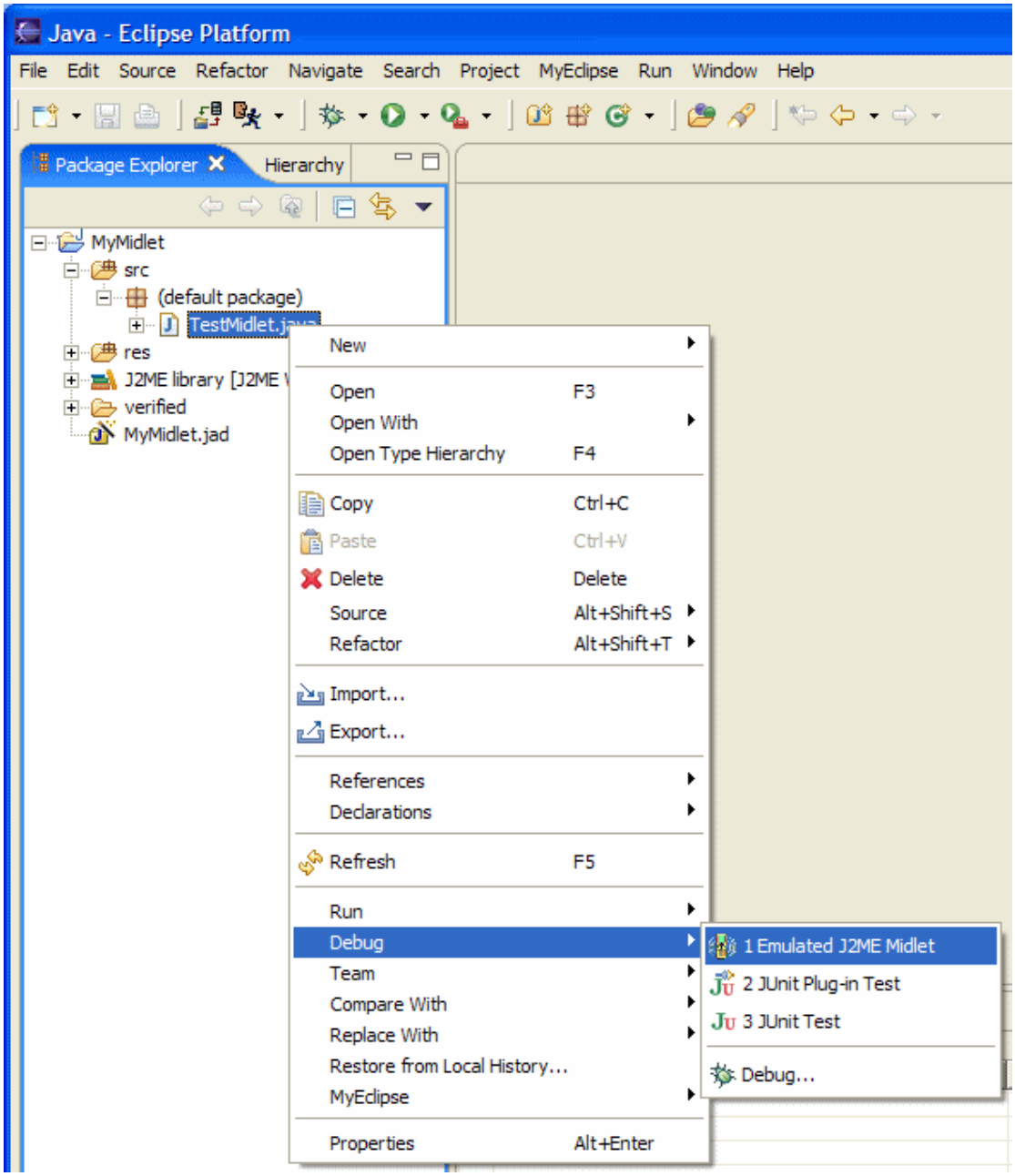
你必须改变一些 Eclipse 的默认设定以调试 MIDlet。在安装文档中说明了需要进行哪些改变。请参考”配置 EclipseME 和 Eclipse”看看你需要进行哪些改变。

7.2. 快速的调试 MIDlet

要对一个 MIDlet 进行快速调试，只要从调试(Debug)项中选择 模拟运行 J2ME MIDlet (Emulated J2ME Midlet)项即可。



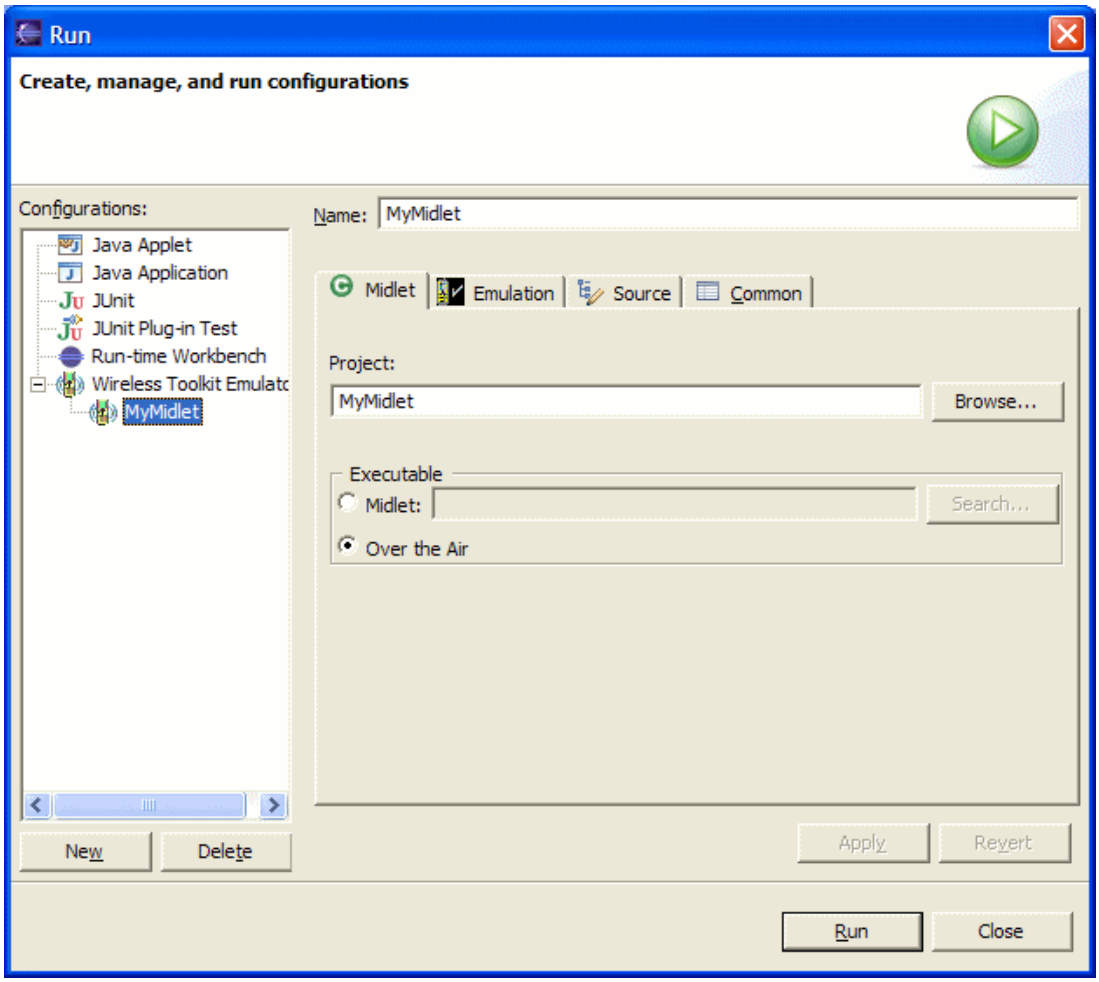
这样 EclipseME 会使用当前的默认设置来自动创建一个启动配置项，并启动调试器。
或者,你也可以使用右键点击你的 MIDlet 类并选择调试(Debug) 项中的模拟运行 J2ME MIDlet (Emulated J2ME Midlet)项， 来使用“根据上下文启动(contextual launch)”的特性。



当你用上述两种方式之一时，就为启动 MIDlet (Midlet launching)模式自动生成了启动配置。

7.3. 手动创建一个启动配置

你可以选择运行(Run)...或调试(Debug)...来手动 创建一个启动配置，点击左面面板中无线工具包模拟 (Wireless Toolkit Emulation)， 然后按新建(New)按钮。



默认情况下，启动配置是按照无线下载模式创建的。

7.4. 在模拟器上调试

从 Eclipse 3.0 开始，就可以在 J2ME 模拟器自带的 KVM 上进行调试了。 “模拟运行 J2ME MIDlet”的启动配置项也可以用于在模拟器中调试 MIDlet。
注意：一般情况下，无法在“无线下载”模式下对 MIDlet 进行调试。

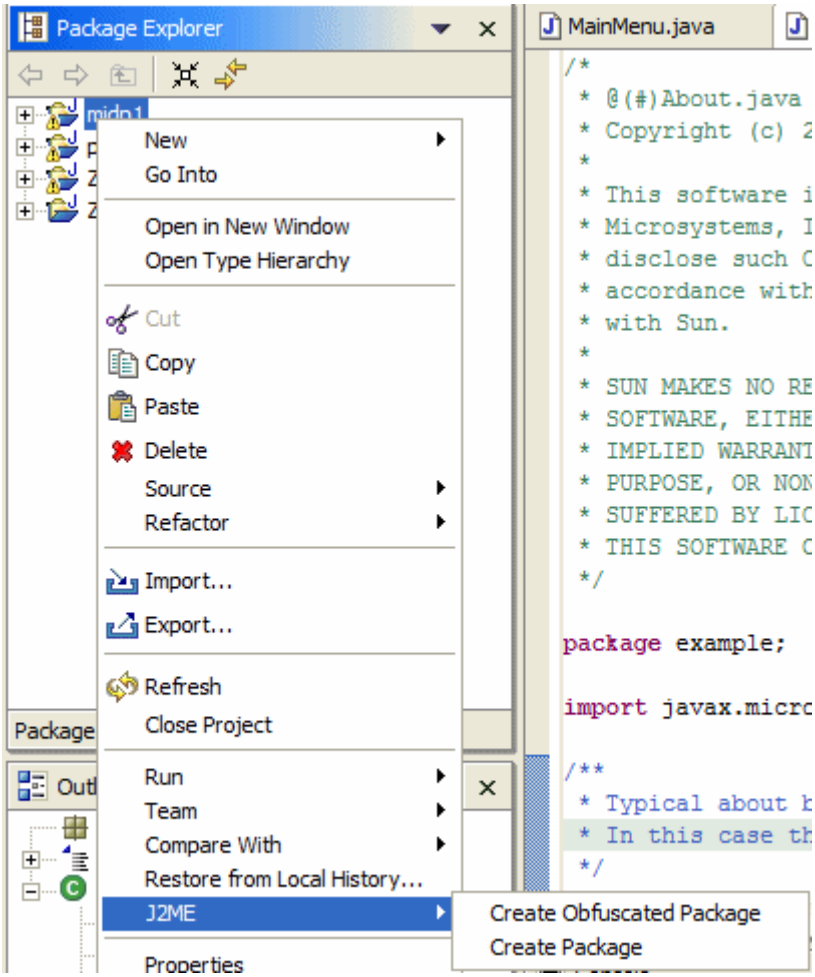
7.5. 有数字签名的 MIDlet

一般情况下，当你使用非 OTA 模式启动时，EclipseME 将尝试使用 verified(经过校验的类)目录下的类来运行模拟器， 而不会进行一次完整的部署过程。但某些模拟器无法在此模式下运行，而需要你的项目被完整地部署。 另外，OTA 启动模式也要求项目被完整部署。
EclipseME 会自动处理这些情况。由于数字签名处理是部署流程的一部分，因此如果你对 MIDlet 进行了数字签名， 那么，根据你对签名密码的设置，在启动过程中 EclipseME 可能会提示你输入密钥库密码和密钥密码。 一般情况下这只会在你把 EclipseME 配置为提示输入密码，并且第一次启动模拟器的时候发生。随后，EclipseME 就会把密码保存在密码中，因此直到你关闭 Eclipse 或改变了工作空间，EclipseME 都不必再次提示你输入密码。
详细情况，请参见关于 MIDlet 套件的数字签名一节。

8. 打包

8.1. 如何把 MIDlet 套件打包

EclipseME 是通过 J2ME MIDlet 套件项目的上下文菜单来提供打包支持的，请参见下面的截图。



8.2. 打包选项

有两种打包选项——创建混淆包(Create Obfuscated Package) 和创建包(Create Package).

- 创建包

如果使用创建包，那么将把 JAD 和 JAR 文件输出到在首选项(Preferences)中配置的部署目录中。部署的 JAR 文件包含 校验过的类文件和资源文件。

- 创建混淆包

如果使用创建混淆包，同样将把 JAD 和 JAR 文件输出到在首选项中配置的部署目录中。EclipseME 会使用在首选项中指定的 Proguard 工具 来混淆部署的 JAR 文件。

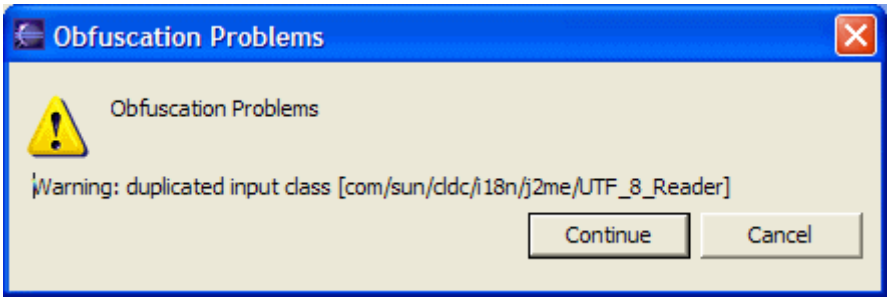
混淆能够对你的 MIDlet 进行一定程度的保护。更重要的是，混淆后的包通常会更小。

为了产生混淆包，需要正确安装 Proguard 工具，并在 混淆首选项(Obfuscation Preferences)

中正确设置其安装目录。 Proguard 是一个免费、开源的工具，可以从 http://proguard.sourceforge.net/ 下载。

- 混淆过程中的错误和警告

在混淆过程中有可能产生警告或错误。这种情况下，会弹出对话框提示是否继续打包或取消。



如果警告或错误可以安全的忽略，你可以选择继续(Continue)， EclipseME 会试图继续创建混淆包。根据问题的类型和严重程度，包有可能无法创建。

8.3.打包输出

部署 JAR 文件

EclipseME 将使用下列信息来把 JAR 文件创建到部署目录中：

在 verified 输出目录中的，经过预校验的 class 文件。、

在资源目录中的资源文件。

根据项目的 JAD 文件内容创建清单文件(manifest)。

部署 JAD 文件

在 J2ME MIDlet 套件项目根目录中的源 JAD 文件将被复制到部署目录中。 请在源 JAD 文件上进行修改，而不要对部署目录中的 JAD 文件进行修改，因为在打包过程中会覆盖掉部署的 JAD 文件。

在 JAD 文件的部署过程中， 会根据刚刚生成的 JAR 文件的实际大小来更新 JAD 文件的 MIDlet-Jar-Size 属性。

混淆包的输出

JAR 和 JAD 文件是在 MIDP 兼容的设备上部署应用所必需的。在此之外，在混淆过程中还产生了一定数量的其它文件。 这些文件和生成的 JAR 文件、JAD 文件一起，被放在部署目录中。

*_base.jar

这个 JAR 文件包含混淆之前的初始包。这个文件被作为混淆处理的源。

*_base_obf.jar

这个 JAR 文件包含混淆后的 class 文件。对这个 jar 文件进行预校验处理就产生了最终的混淆并校验过的 JAR 文件。

pro_map.txt

这个文件包含了类中原名和混淆后的名字的对应关系。 利用这个文件，就可以使用 Proguard 的 ReTrace 命令， 来根据混淆后的追踪输出(stack trace)来重建原始的追踪输出。

pro_seeds.txt

这个文件列出了作为混淆种子(seed)的 MIDlet 子类。

proguard.cfg

这个文件包含 Proguard 的混淆参数设置。

8.4.使用 Antenna 打包

先决条件：
Antenna库(可从http://antenna.sourceforge.net/获得)必须在基本J2ME首选项(Basic J2ME Preferences)中正确配置。
在平台组件(Platform Components)中正确配置了 EclipseME 支持的 Sun 无线工具包
如果这些先决条件未获满足，就会弹出一个错误信息对话框。
从 J2ME 项目菜单选择“导出 Antenna 构建文件(Export Antenna Build Files)”来导出 Antenna 的构建文件。 会自动产生下列几个文件：
build.xml - 用来进行标准 Ant 构建流程的根构建文件。 这个文件是包含 eclipseme-build.xml 文件的骨架文件。 这个文件可以修改，它不会被接下来的导出操作覆盖掉。
eclipseme-build.xml - 这个构建文件处理对 Antenna 库的调用并包含所有与导出的 Eclpse 项目相关的类路径(classpath)信息。 这个文件不能被修改，因为它会被后面的导出操作重写。只要项目的类路径(classpath)更新了，这个文件就会被重新生成。
eclipseme-build.properties - 这个文件包含了一些跟构建相关的位置和首选项定义。这个文件也不能被修改，它也会被后面的导出操作重写。 在此文件的同一路径内创建一个 user-build.properties 文件就可以覆盖这个文件里的属性。 user-build.properties 中定义的属性值会覆盖掉 eclipseme-build.properties 中的同名属性值。

8.5.在打包过程中进行数字签名

从 EclipseME 0.7.0 版本开始，你可以在打包流程中对你的 MIDlet 套件自动进行数字签名。经过的签名的 MIDlet 套件可以运行在“受信任的第三方(trusted third party)”保护域中，而不是在“不被信任的(untrusted)”保护域中。如果你的 MIDlet 需要访问被保护的特性，那么进行数字签名就很重要。
为了在打包流程中对 MIDlet 套件进行数字签名，你要在项目属性页中选中 签署项目(Sign project)复选框。 你需要提供密钥库文件的位置，密钥库中密钥的别名，另外可选的，你也可以提供密钥库和密钥的密码。

9. 高级话题

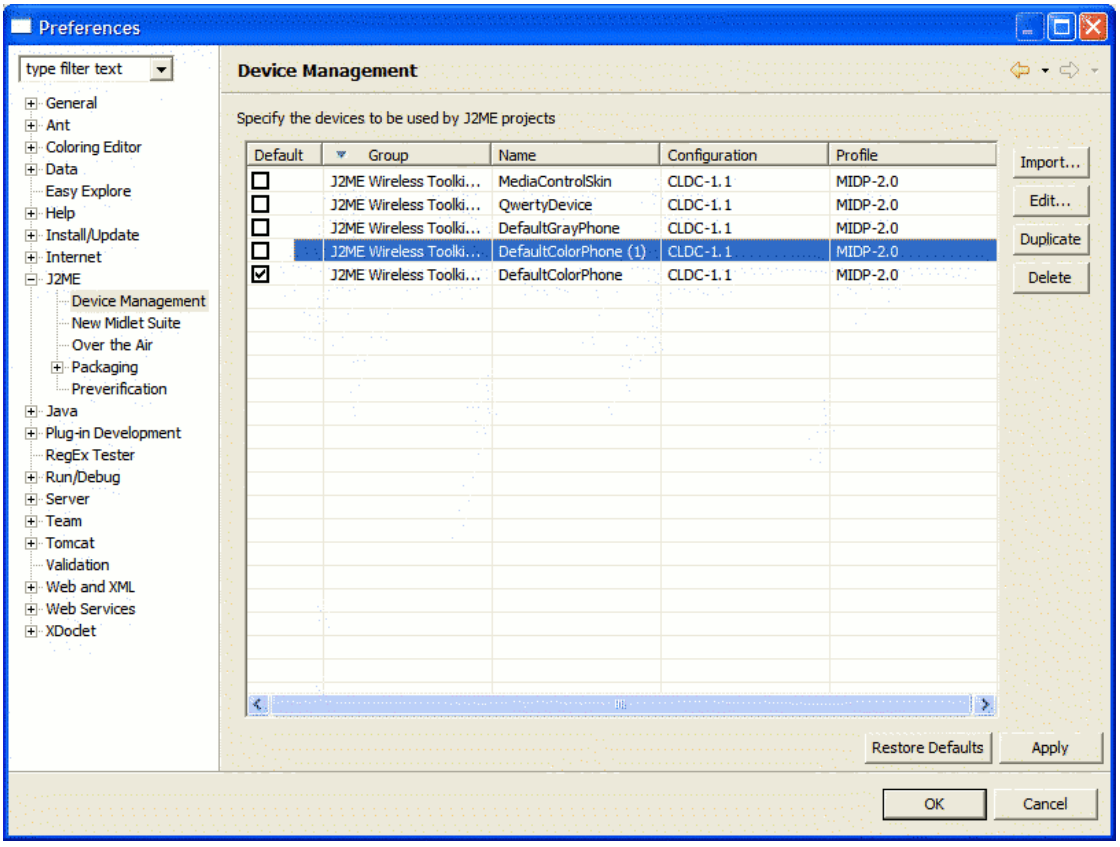
9.1. 设备管理

- 设备管理首选项

为了使用 EclipseME，你必须至少配置一种设备定义。 EclipseME 中的设备管理功能允许你有效控制对 MIDlet 套件和模拟器启动适用的设备定义。 这些设备定义是使用 Eclipse 的首选项对话框进行管理的。按下面的步骤进行设备管理：

1. 选择 Eclipse 的窗口(Window)菜单中的首选项(Preferences)。
2. 展开左面面板的 J2ME 选项分支并点击设备管理项。

右面的设备管理面板显示了当前已被 EclipseME 识别的所有设备。 在这个面板中，可以导入新设备，重新定义当前设备，以及删除设备定义。

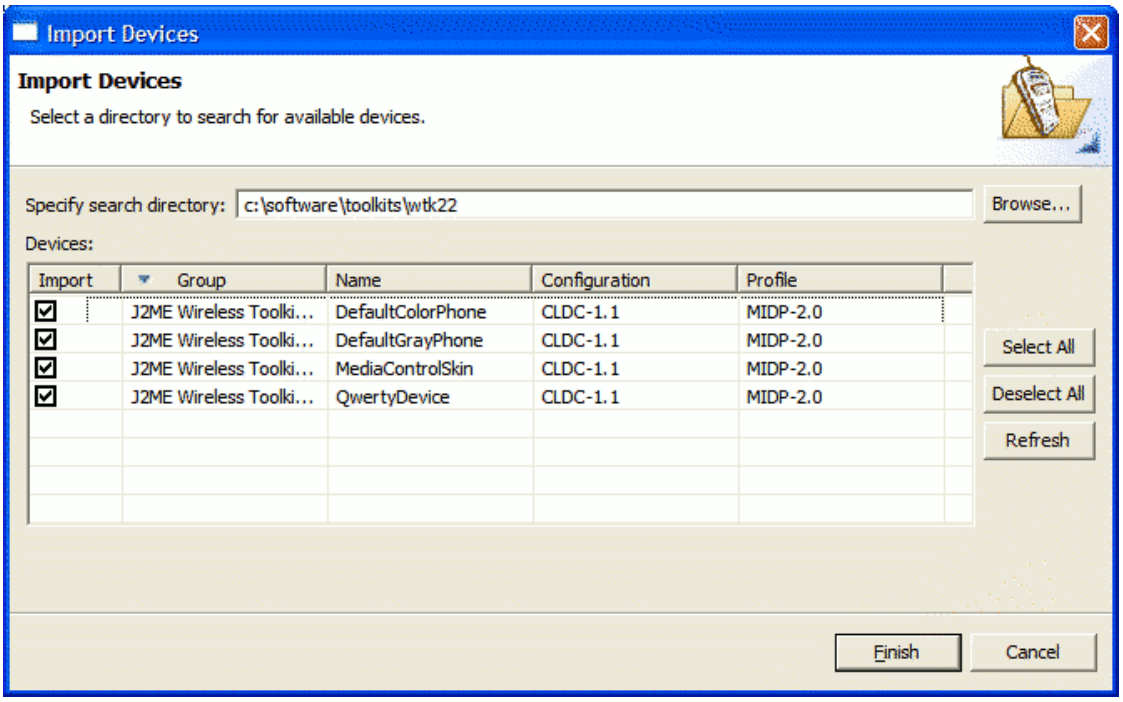


可以在默认(Default)列中选中一个设备作为默认设备。 当创建新的 MIDlet 套件和新的模拟器启动配置的时候会首先使用默认设备。

- 导入设备

EclipseME 可默认识别数种不同类型的无线工具包和其相关目录结构。 这些设备可以被自动导入 EclipseME，用于定义项目和启动模拟器。

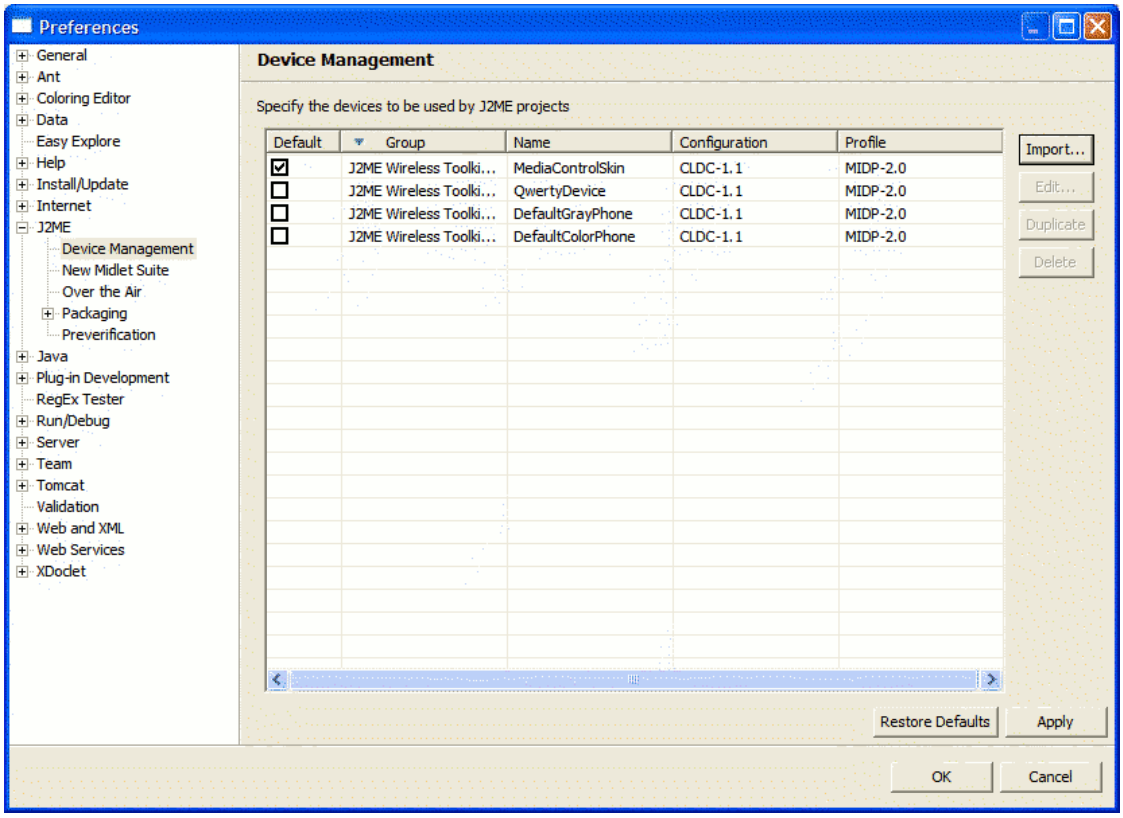
1. 选择导入(Import)...
2. 在接下来的对话框中，选择已知设备定义的根目录来进行搜索。



当你离开搜索目录文本框或者按下了刷新(Refresh)按钮， EclipseME 就会开始在给定的目录和其所有子目录中查找设备。从 EclipseME 1.5.0 版本开始， 不再需要给定“精确”的目录来进行导入，因为只要它们存在于给定的目录中， EclipseME 就应该可以定位到它们。只要找到设备，EclipseME 就会把它显示出来。 如果你希望停止搜索，只要按下取消(Cancel)按钮。 搜索结束以后，使用复选框选中你要导入的设备定义。 最后，选择完成(Finish)按钮来结束导入流程。

如果 EclipseME 未能如你所愿找到设备定义，有可能是 EclipseME 暂时还不支持它。出现这种情况， 请向我们提交改进需求(enhancement request)，来要求我们加入对这种 WTK 的支持。

当你成功的添加了设备，你会在设备管理首选项中看到导入的设备。



选择完成(Finish)以保存定义。

- 复制设备定义

在某些情况下，你可能发现导入的设备定义接近你的需求，但需要进行少许修改。 这种情况下，你可以使用复制(Duplicate)按钮来创建选定设备定义的 一个精确的拷贝。可以对这份复制的设备定义进行必要的修改，而不会影响原版设备定义。

注意：并不是一定要在修改之前进行复制。 如果你对导入的设备定义进行了修改并希望恢复原版，只要简单的删除设备定义并重新导入即可。

- 编辑设备定义

EclipseME 试图识别各种开发者用得到的无线工具包。 但不幸的是，保持跟进不同的厂商和和他们五花八门的工具包结构是非常困难的。即使 EclipseME 能够识别和导入一个设备定义，它仍有可能是不正确的，或不完全符合你的要求。 这时可以选择编辑(Edit)...按钮以打开编辑对话框。

这个有若干个标签面板的对话框提供了对设备定义的控制，包括模拟器运行，启动命令和设备的相关库。 当编辑对话框打开的时候，当前选定的设备定义内容就自动填充到对话框中。所有的修改完成之后，点击对话框的确定(OK)按钮， 然后点击设备管理首选项对话框的确定(OK)按钮。

- 基本属性

编辑对话框的第一个标签面板显示了设备的许多属性定义。所有这些属性在一起即组成了 EclipseME 的必要环境，因此进行修改的时候请务必小心。对此面板上各种属性进行配置的进一步信息， 请参见编辑设备基本属性一节。

- 库

编辑对话框的库(Libraries)面板允许对设备定义的库进行 选择和配置。应用本面板可以控制增减设备定义的功能特性。可以用这种方法来限制可用的 API，以创建 更通用的项目。请

参见编辑设备的相关库一节来了解详细信息。

- 高级属性

编辑对话框的属性(Properties)标签面板显示了当前选定设备的属性定义。 这些属性目前还不可以修改。

9.2.把 JAR 文件添加到 MIDlet 套件

- JAR 文件类型

从一个 J2ME 项目的观点来看，有两大类可以被添加到 MIDlet 套件项目的 JAR 文件。 理解这两类 JAR 文件的区别是很重要的，因为它们必须被区别对待才能得到你想获得的结果。 这两类是：

- 硬件相关 JAR 文件

  这类 JAR 文件向 MIDlet 提供与一些硬件功能交互的支持，它们是安装 MIDlet 的目标手机或其它移动设备硬件的一部分。 这类文件的一个例子是 Bluetooth™（蓝牙）接口类。

  这类 JAR 文件是由设备制造商提供的，通常作为无线工具包的一部分或其扩展包(add-on)的形式提供。 这类 JAR 文件单独存在的意义在于，使 J2ME 开发者可以使用它们对 MIDlet 进行正确的编译， 且在部署后正确引用它们。你不需要把这些硬件相关 JAR 中的 class 安装到移动设备中去—— 因为它们已经在那里了。
- 应用相关 JAR 文件

  这类 JAR 文件包含由你或者其他 J2ME 开发者编写的、将直接作为 MIDlet 的一部分而被打包部署到移动设备的类文件。 例如，一个你用来处理与服务器之间交换数据的 XML 解析库。

  这类 JAR 文件一般是从设备制造商之外的地方获取的。当然，如果你把你的代码打包成了 JAR，那它也属于这类。

对它们进行区别的原因在于，EclipseME 要对它们中的 class 文件进行不同的处理。 对于硬件相关 JAR 文件，EclipseME 要保证这些文件中的类在构建过程中是类路径(classpath)的一部分， 这样你的 MIDlet 才能正确编译，但这些类绝不能被打包到你的 MIDlet 部署包中去。 因为这样做相当于试图覆盖移动设备中的硬件实现类的功能。这或许根本无法工作，因为首先，这种类文件很多都 包含本地代码(native code)，而不能在 MIDlet 运行环境的“沙盒(sandbox)”中工作。 其次，一个包含本地代码的 MIDlet 在部署流程中应该是无法通过预校验的。最后，即使上面的情况都没有发生， 只要是自律的 MIDlet 容器都不会允许 MIDlet 覆盖系统类——它会认为 MIDlet 是无效的而拒绝执行。

另一方面，对于应用相关 JAR 文件，EclipseME 则需要抽取其中的所有类和资源文件等，把它们和 MIDlet 套件项目中的相应项目一起，打包到最后的部署包中。J2ME 设备中并不存在 J2SE 和 J2EE 系统中的 类路径(classpath)概念。一个 J2ME 套件只是由两个文件组成——一个 JAR 文件和一个 JAD 文件。 因此所有你的 MIDlet 要用到的类和资源都必须被打包成一个单独的 JAR 文件。

EclipseME 根据你把文件加入系统的方式以及（隐含的）jar 文件的相关“导出(export)”设定来区分其分类归属。 因此下面的工作对你很重要：

1. 决定 JAR 文件的归属分类，然后
2. 用正确的方式把它加入系统。

- 添加硬件相关 JAR 文件

有两种添加硬件相关 JAR 的方法。可以在设备管理首选项面板中添加，也可以直接添加到项目，但不进行导出。

- 是用设备定义来添加硬件相关 JAR 文件

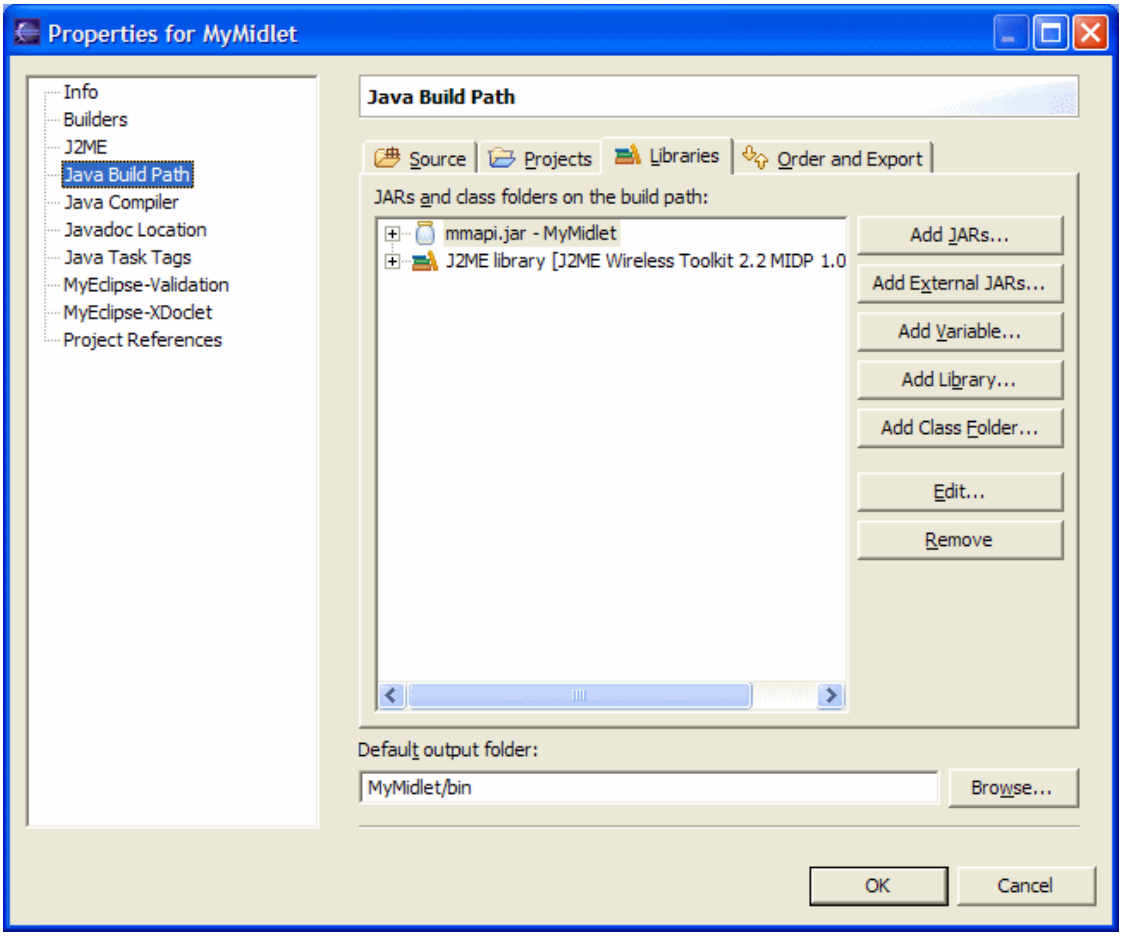
硬件相关 JAR 文件可以从设备管理首选项中作为一个新库来添加。 大部分情况下，EclipseME 能够从你的无线工具包中获取所有库定义的相关信息。 如果没有的话，那么你需要首先向设备定义中添加一个新库。 最后，修改你的 MIDlet 套件项目的属性来使用新添加的或更新过的设备定义。

以上步骤结束之后，JAR 文件就包含到你的项目的构建路径中了，这样编译器可以找到相关类的定义， 但 EclipseME 不会试图把这些类添加到你的部署 JAR 文件中去—— 因为 EclipseME 知道：这个 JAR 文件现在是无线工具包的一部分，仅用于类引用的目的。

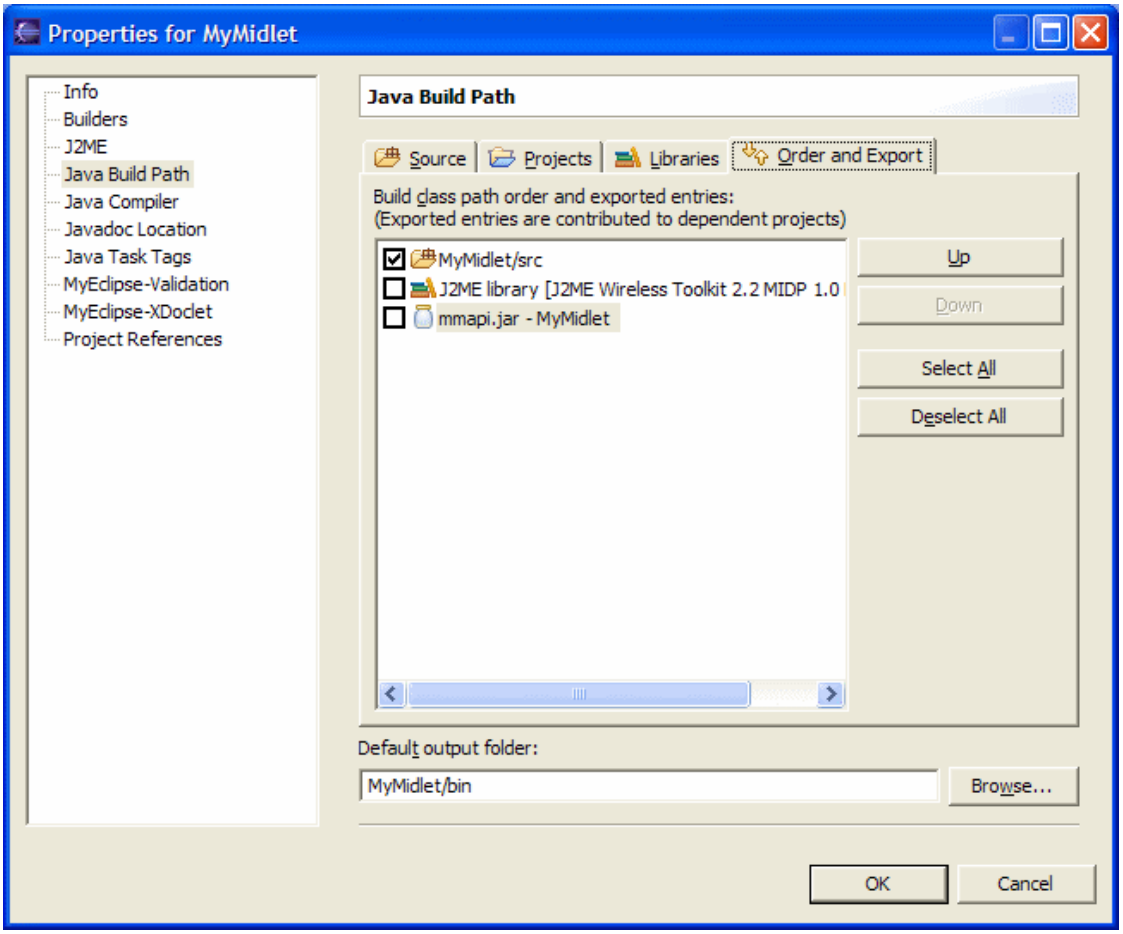
如果你试图在模拟器运行 MIDlet 的时候，特别是在 OTA 模式中，你得到了非法类(illegal classes) 的错误提示，这可能是一个信号：你把硬件相关 JAR 错误的按照应用相关 JAR 的方式加入了系统。

- 直接添加到项目

在 EclipseME 0.9.4 版本之前，上面的方法是添加硬件相关 JAR 到 EclipseME 的唯一方案。 从 0.9.4 版本开始，硬件相关 JAR 文件还可以直接在项目属性的“Java 构建路径”中的“库”面板中直接添加。 比如，在下面的项目中，mmapi.jar 已经被添加到了项目中。



但是，如果你按此方法添加 JAR 文件，需要你高度注意的是：在“排序和导出(Order and Export”面板中， 千万不要选中该 JAR 文件旁边的复选框，就像下面这样：



如果选中了复选框，那么 EclipseME 会试图对 mmapi.jar 中的内容进行 预校验并把其内容包含在输出的部署 JAR 文件中，这显然是错误的。

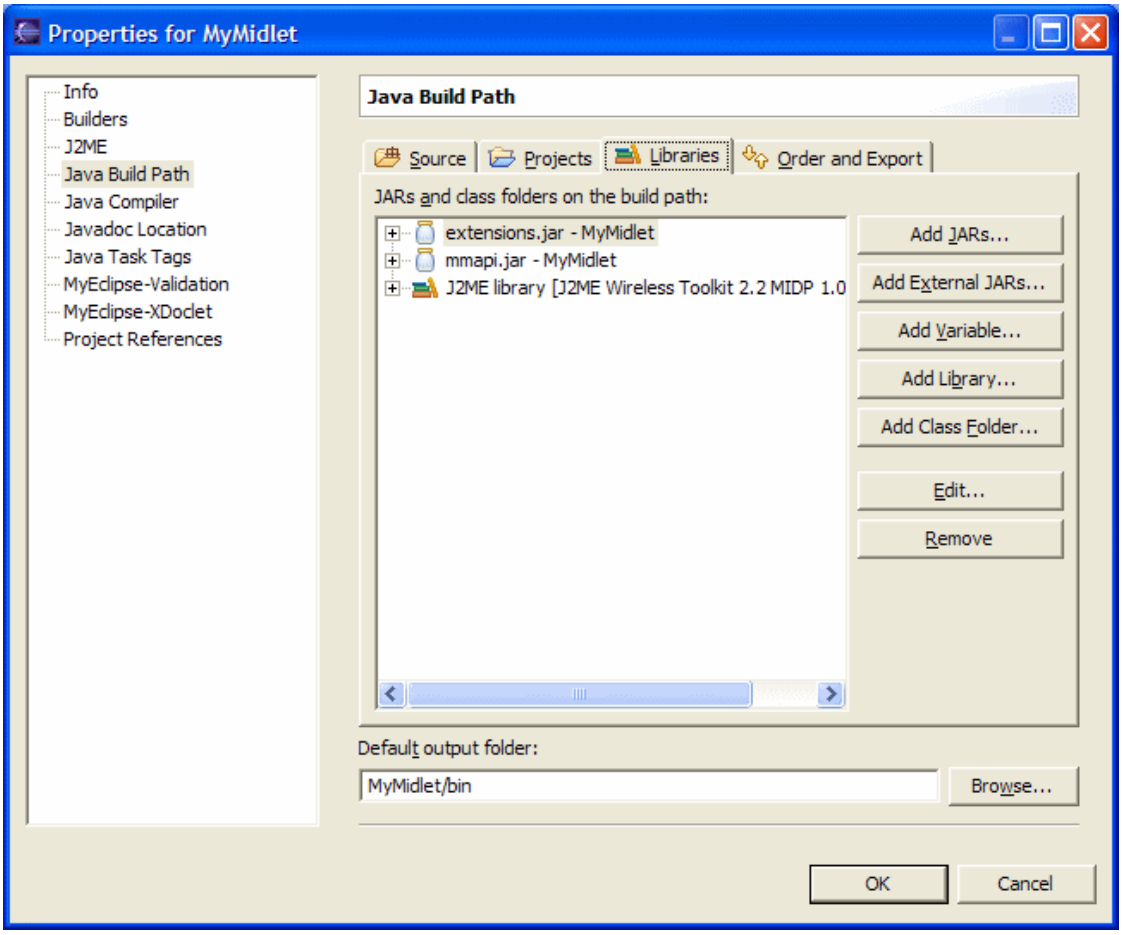
- 哪个方法更好？

在 EclipseME 0.9.4 之前，你只有一种方法添加硬件相关 JAR，所以事情很简单。从 0.9.4 版本开始， 你有了两个选择，那么哪个更好些呢？

EclipseME 的开发者认为，在绝大部分情况下，最好使用第一种方案——使用设备定义来添加硬件相关 JAR 文件。 这个方法使得这个改变在不同项目中获得重用， 而且更好的把硬件相关 JAR 文件和你的项目所使用的硬件设备相联系。但如果因为某些原因你无法使用这个方法，那么第二种方法——添加不进行导出的 JAR 文件到项目——你也可以选用。

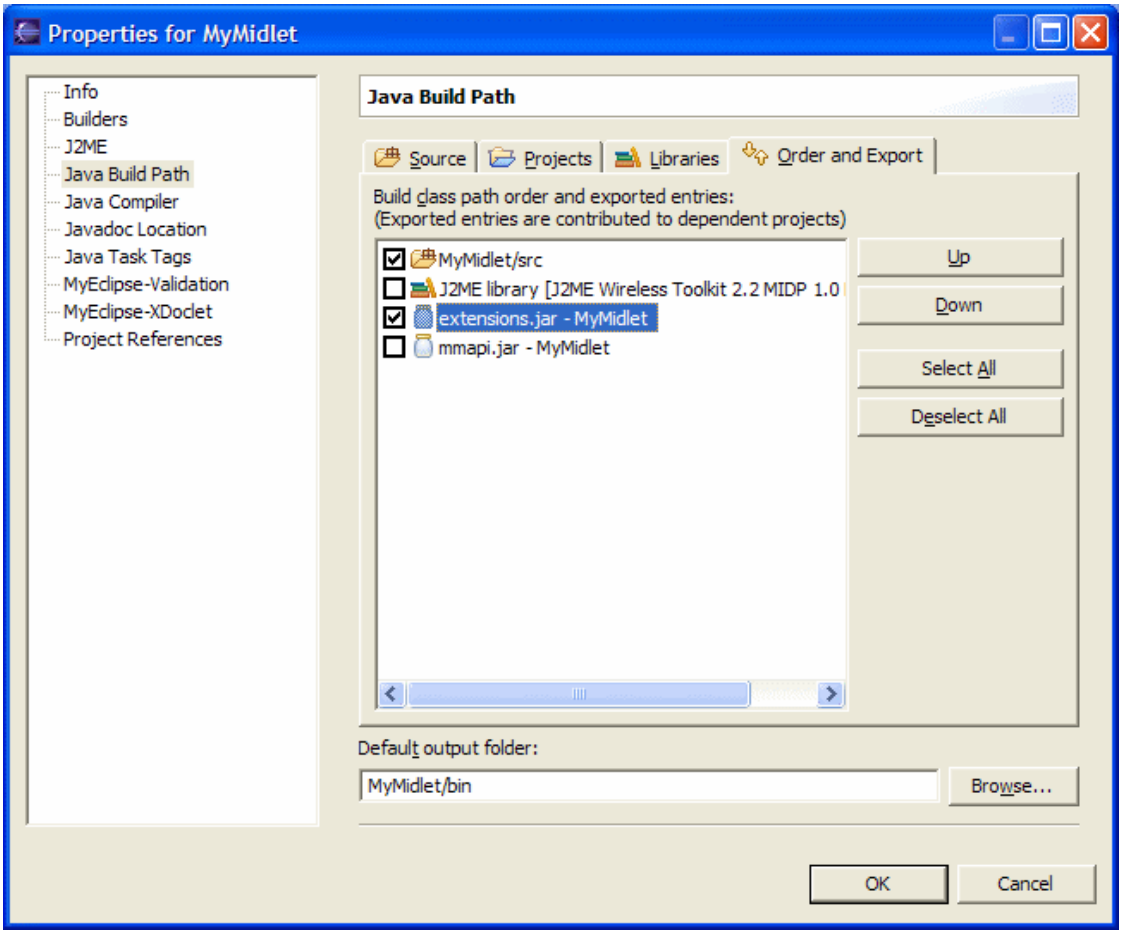
- 添加应用相关 JAR 文件

应用相关 JAR 文件必须像非 J2ME 项目中的 JAR 文件那样， 被加入到项目的构建路径中。选择项目菜单中的 属性(Properties)菜单项，点击左边面板的 Java 构建路径(Java Build Path)项，然后在对话框的右边面板 中加入 JAR 文件（或类文件目录）。



就像这样，在上面的例子中，除了已经加入的硬件相关 JAR 文件 mmapi.jar 之外， 我们正在为项目加入一个应用相关 JAR 文件 extensions.jar。

从 EclipseME 0.9.4 版本开始，你必须在 Java 构建路径中的 “排序和导出”面板中，选中所有的应用相关 JAR 文件旁边的复选框，像下面这样：



完成之后，JAR 文件或类文件目录中的 class 文件就成为你的项目构建路径的一部分， 而且会被打包到项目的输出部署 JAR 文件中。 就像上面的例子中，EclipseME 会收集 extensions.jar 中的所有类和其它资源，进行预校验，然后把它们作为你的 MIDlet 的一部分进行打包部署。 但是，EclipseME 不会如此处理 mmapi.jar， 因为在“排序和导出”中，它旁边的复选框是空的。mmapi.jar 中的资源 在编译时是可用的，但不会被打包部署。

如果你试图在模拟器运行 MIDlet 的时候，特别是在 OTA 模式中， 你得到了 ClassNotFoundException（未找到类）异常， 这是个信号，表示相关的类未被正确打包到你的部署 JAR 文件中。 通常这意味着你没有正确的把应用相关 JAR 文件添加到项目的 Java 构建路径中， 或者是忘记选中“导出”复选框了。

- 模拟器说明

如果你正在使用一些特殊特性，比如蓝牙、视频等，你可能会处于这样一种境地： 你按照上面的方法正确的添加了硬件相关 JAR，你的应用程序也可以正确的编译， 但是在模拟器中运行的时候，硬件功能却似乎并不工作。

这种情况下唯一的可能性就是，这并非 EclipseME 的问题——而很可能是 J2ME 模拟器的问题。在很多情况下，无线工具包的模拟器对硬件特性的支持仅限于正确载入 MIDlet， 而不包含对物理设备硬件特性的真实模拟。不幸的是，对这种情况我们也无能为力。

9.3. 为 WTK 库连接文档(Javadoc)或源代码

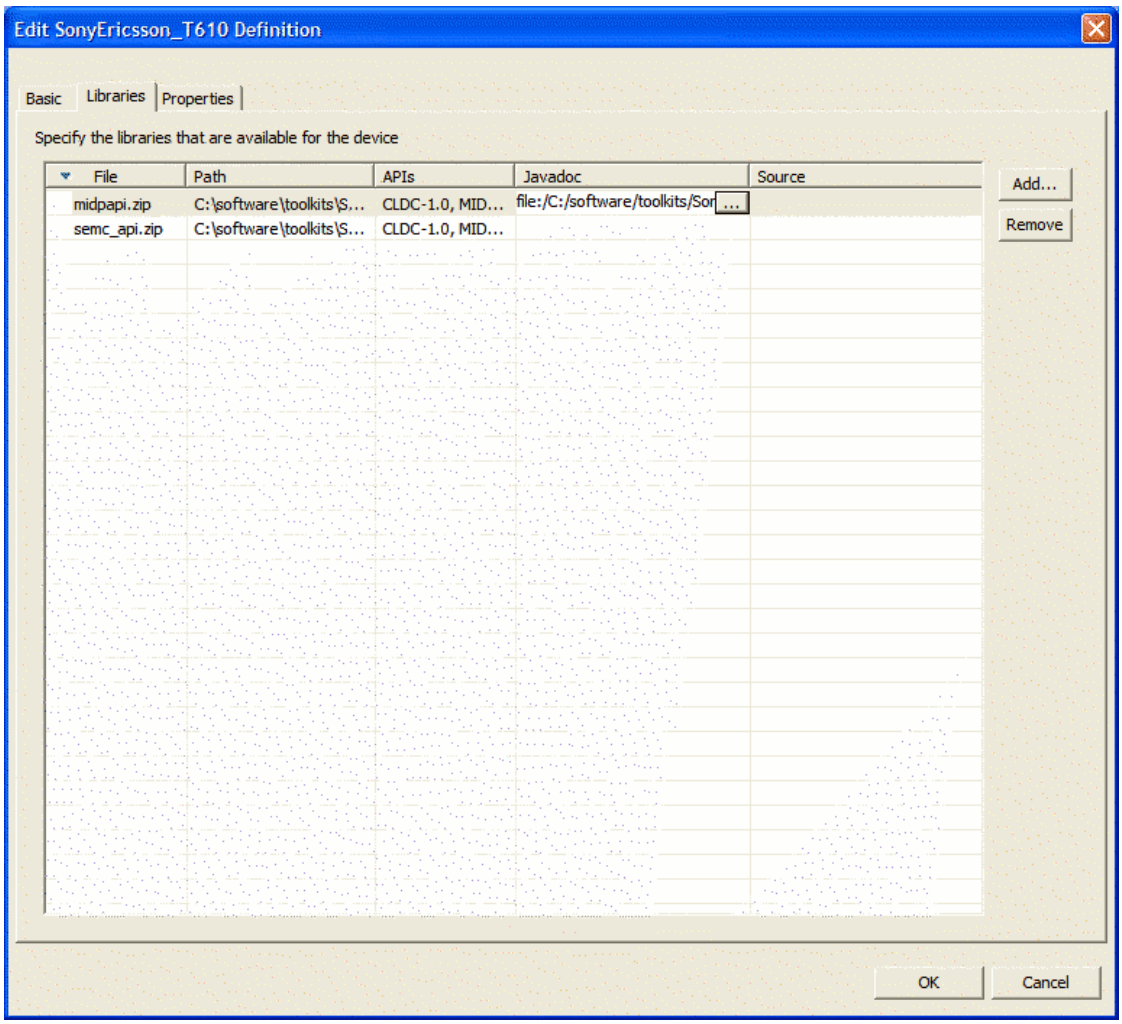
从 EclipseME 0.9.0 版本开始，你可以向无线工具包(WTK)的库添加 Javadoc 或源代码的连接。这样你可以进行源码级别的调试（如果编译的 WTK 中包含调试信息的话）， 并且可以使用

Eclipse 的上下文敏感 API 帮助功能。

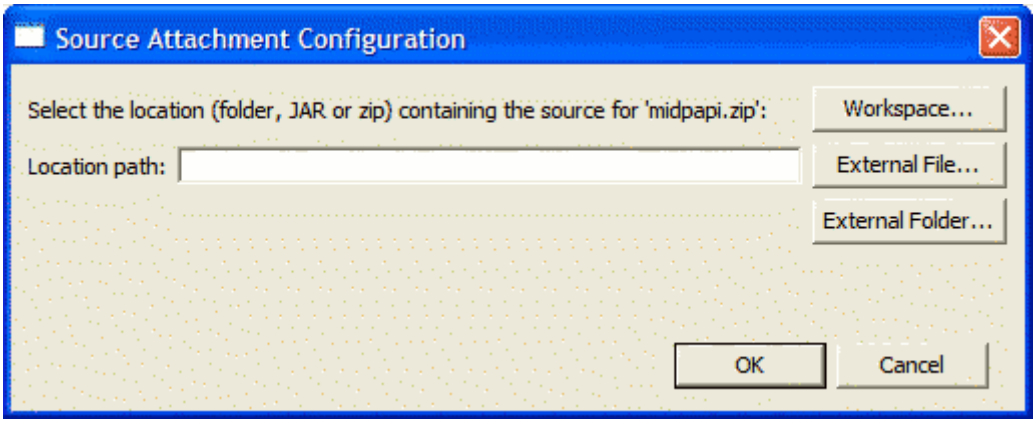
- 为 WTK 库连接源码或 Javadoc

为了向 WTK 组件连接 Javadoc，请进行如下步骤：

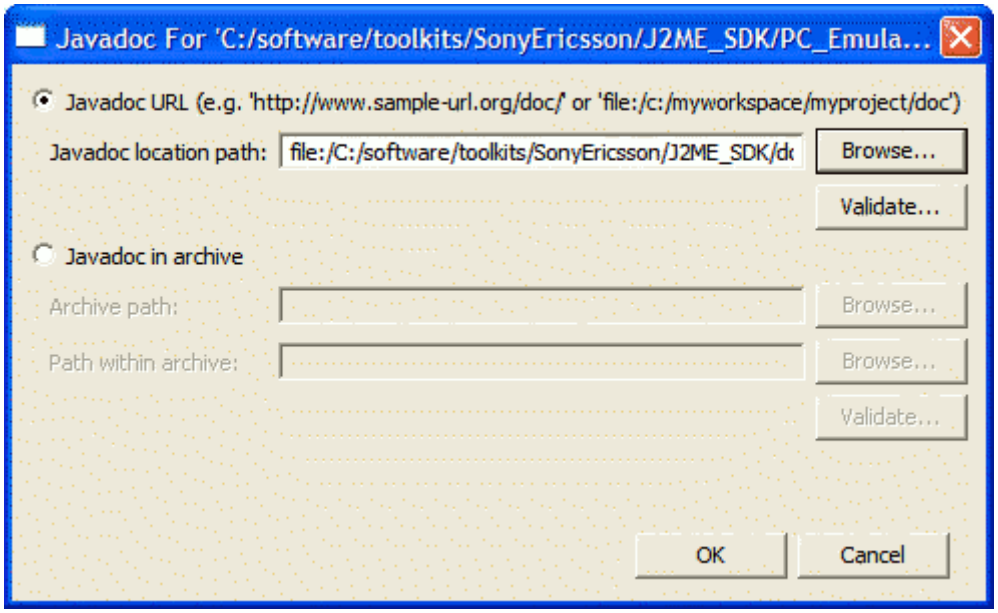
1. 选择窗口菜单的首选项， 点开左面的 J2ME 分支， 然后点击设备管理(Device Management)，打开设备管理器窗口。
2. 选择你要添加源码和/或 Javadoc 的设备，然后点击编辑(Edit)...按钮。
3. 点击 Javadoc 或源码(Source)单元格以进入编辑模式。



点击单元格编辑器的按钮以打开适当的编辑器对话框。 连接源代码，请使用弹出的源代码连接配置对话框。



连接 Javadoc，请使用弹出的 Javadoc 连接配置对话框。



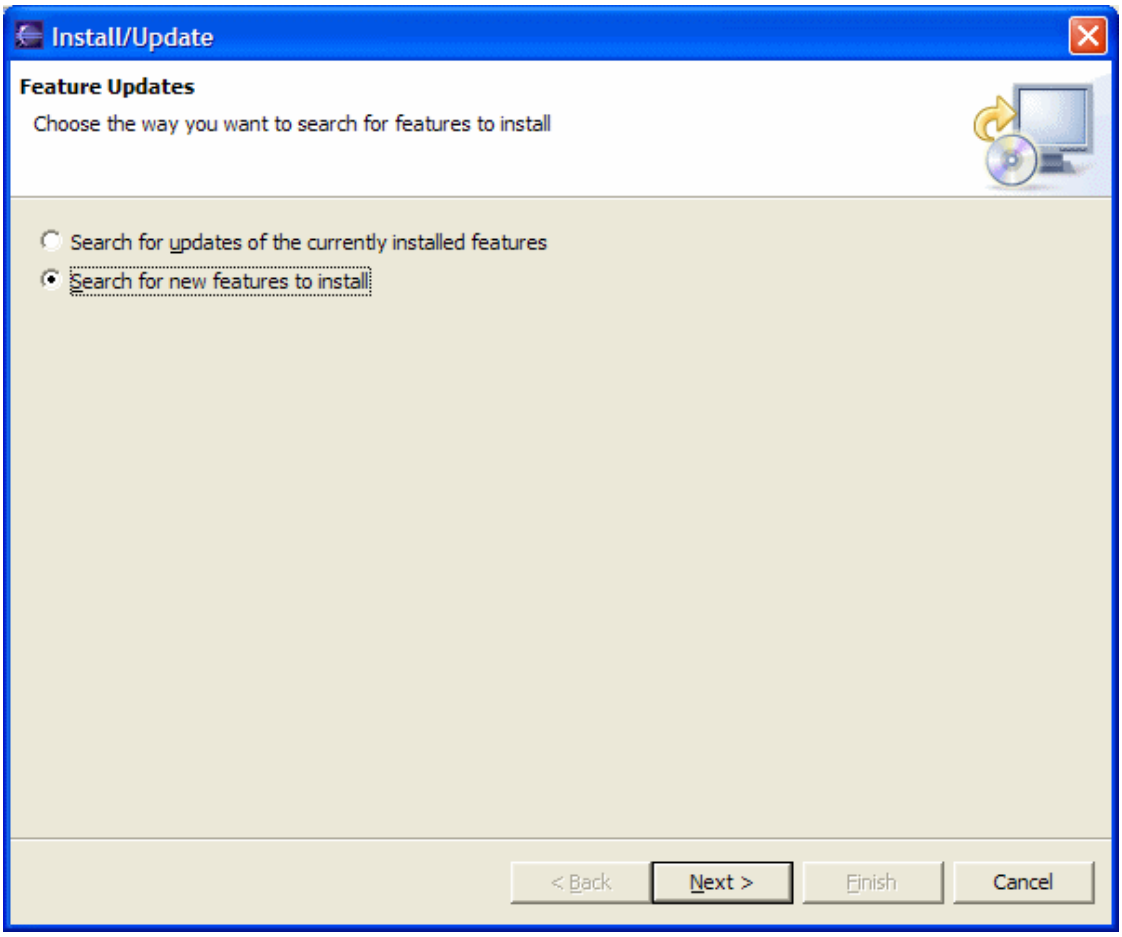
注意并非所有的无线工具包的 JAR 文件都是使用包含调试信息的方式编译的。 因此，即使你已经给 JAR 文件和源代码建立连接，你可能仍然无法在调试中跳入 WTK 源代码。

10. 更新 EclipseME

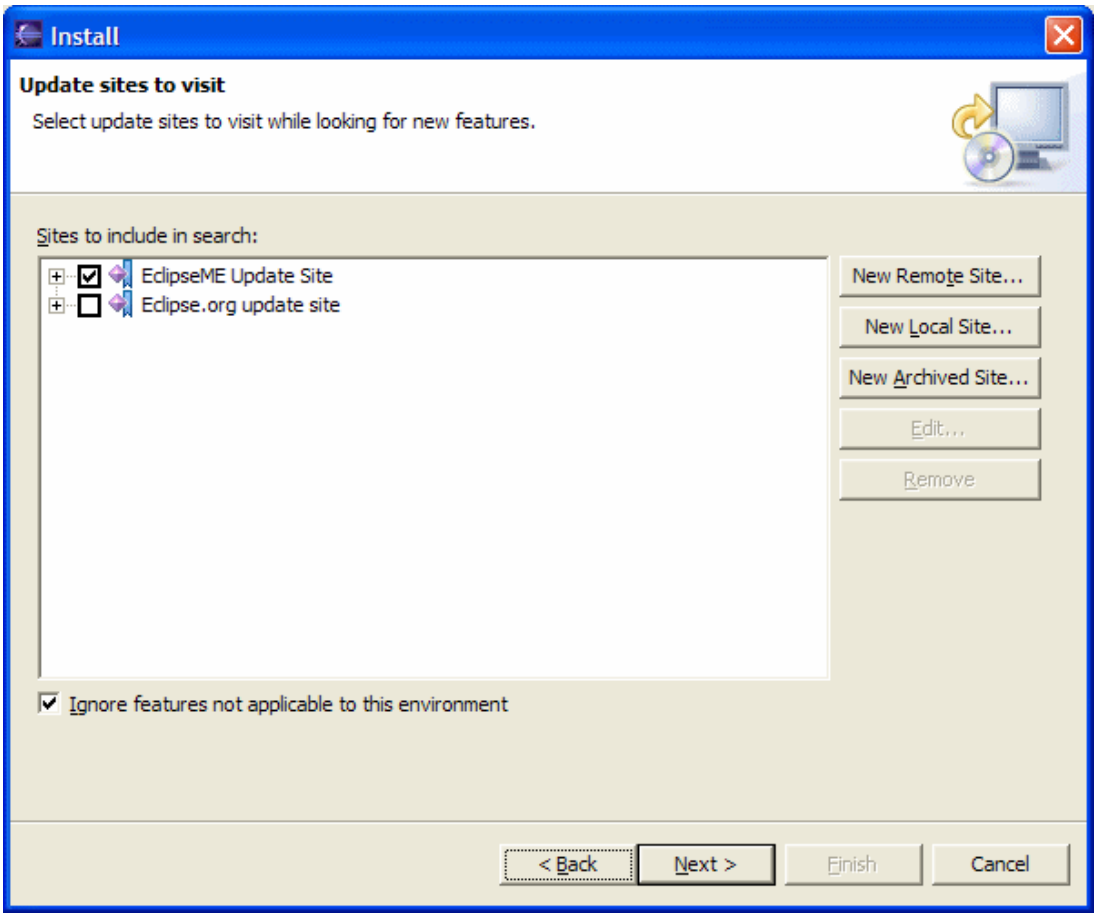
Eclipse 实现了一个版本更新系统，使得你可以不必卸载旧版本就可以更新功能和插件。 因此，更新 EclipseME 的流程相当简单。

- 如果你通过 EclipseME 更新站点进行的首次安装

从 Eclipse 的帮助菜单中选择软件更新， 然后选择查找并更新... 你会看到下面的对话框：



选中搜索要安装的新功能部件(Search for new features to install). 按下一步按钮。在安装对话框中会列出你以前配置过的更新站点。



请确保在 EclipseME 更新站点(EclipseME update site)左面的复选框已被勾选， 然后按下下一步按钮。从这一步开始，后面的流程就和前面说过的安装流程一致了。

注意，目前还不能用搜索当前已安装功能部件的更新(Search for updates of currently installed features)这个功能来更新 EclipseME。你必须依照上面的流程直接从 SourceForge 站点来进行更新。 我们期望在将来可以修正这个问题。

- 如果你使用下载的安装包进行的首次安装

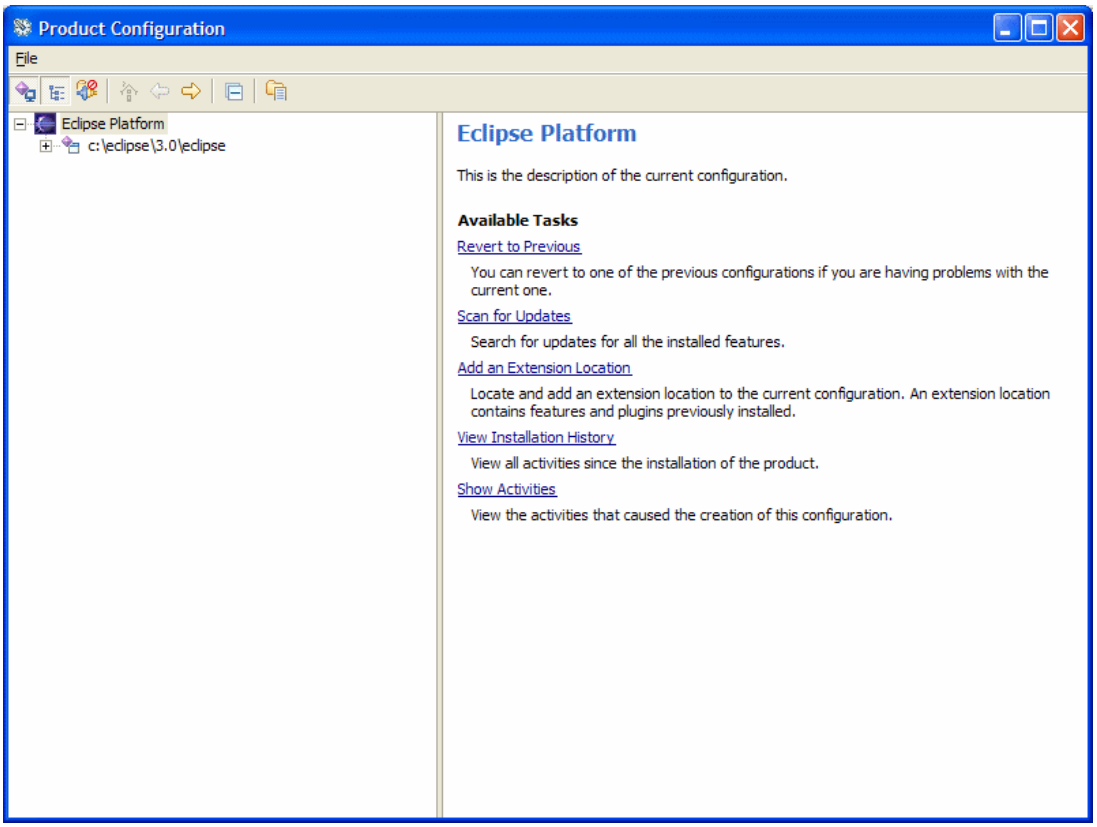
更新流程和初次安装的流程是一致的。 只要下载新的安装包文件，并使用它重新走一遍安装流程就可以了。 如果你愿意，在安装新文件之前，你可以使用安装对话框上的除去(Remove)按钮来卸载掉旧版本。

11. 卸载 EclipseME 插件

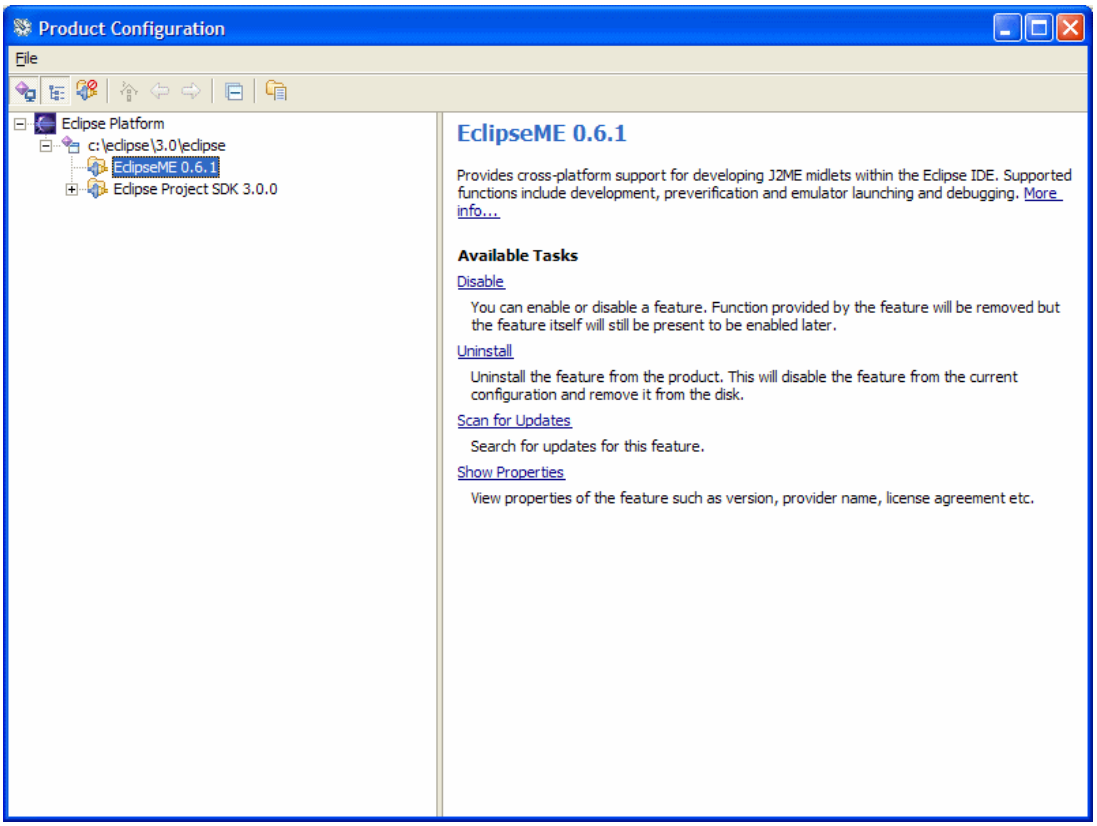
很显然我们当然希望你认为 EclipseME 是如此的有用以至于你永远也不会放弃它。 不过，如果你真的认为你需要卸载它，或者你想进行一次完整的“干净的”安装， 你可以按照下面的步骤把 EclipseME 从 Eclipse 中卸载：

- “一般”方式

删除 EclipseME 的最简单方式就是使用 Eclipse 的配置管理器。
从 Eclipse 的帮助菜单中选择软件更新 中的管理配置(Manage Configuration)... 你会看到下面的对话框：



展开左边面板的 Eclipse 安装目录分支，然后点击 EclipseME 的分支。这时，对话框看上去应类似这样：



点击右面的反安装(Uninstall)链接。

Eclipse 会请求你确认是否卸载 EclipseME。 回答“是”，接着 EclipseME 就会被卸载了。注意卸载后 Eclipse 必须重启，以更新其功能和插件的元数据。

如果你安装了 EclipseME 的可选功能模块（比如 Siemens(西门子)特性模块)， Eclipse 会需要你在卸载它们之前先卸载 EclipseME 本身。 这种情况下，在卸载这些模块和 EclipseME 本身的过程中间没必要重启。

注意用这种方法卸载 EclipseME 只会从 Eclipse 安装目录中删除当前版本的 EclipseME 特性和插件。 如果你的 EclipseME 是从旧版本升级而来，那么旧版本的插件和特性目录会仍然留在 Eclipse 的目录中， 你可以用下面的“暴力”方式除去它们。

- “暴力”方式

如果“一般”方式不奏效，或者你希望卸载的绝对干净，那么你总是可以使用“暴力”方式来删除 EclipseME：

1. 关闭所有运行着的 Eclipse。
2. 打开 Eclipse 安装目录中的 features 目录。
3. 删除所有名字以"eclipseme"开头的目录（比如 eclipseme.feature_0.6.1)。
4. 打开 Eclipse 安装目录中的 plugins 目录。
5. 删除所有名字以"eclipseme"开头的目录（比如 eclipseme.core_0.6.1，eclipseme.docs_0.6.1 等等)。
6. 使用-clean 命令行参数来启动 Eclipse。 这会强制 Eclipse 重新扫描其功能和插件列表以确保对 Eclipse 的元数据进行适当的更新。

- 删除 EclipseME 在项目中留下的痕迹

如果你还希望删除 EclipseME 在项目中留下的所有痕迹，你需要在 Eclipse 之外做下面的工作：

EclipseME 会在每个 J2ME 项目的根目录创建一个叫.eclipseme 的文件。 删除这个文件。

EclipseME 会在每个 Eclipse 项目的.project 文件中添加一个构建命令， 和一个"nature"。一个典型的 Eclipse 项目是类似这样的：

```
<xml version="1.0" encoding="UTF-8"?>
<projectDescription>
    <name>PaperClick Java Go Window</name>
    <comment></comment>
    <projects>
    </projects>
    <buildSpec>
        <buildCommand>
            <name>org.eclipse.jdt.core.javabuilder</name>
            <arguments>
            </arguments>
        </buildCommand>
        <buildCommand>
            <name>eclipseme.core.preverifier</name>
            <arguments>
            </arguments>
        </buildCommand>
    </buildSpec>
    <natures>
        <nature>org.eclipse.jdt.core.javanature</nature>
        <nature>eclipseme.core.nature</nature>
    </natures>
</projectDescription>
```

请删除有下划线的部分。

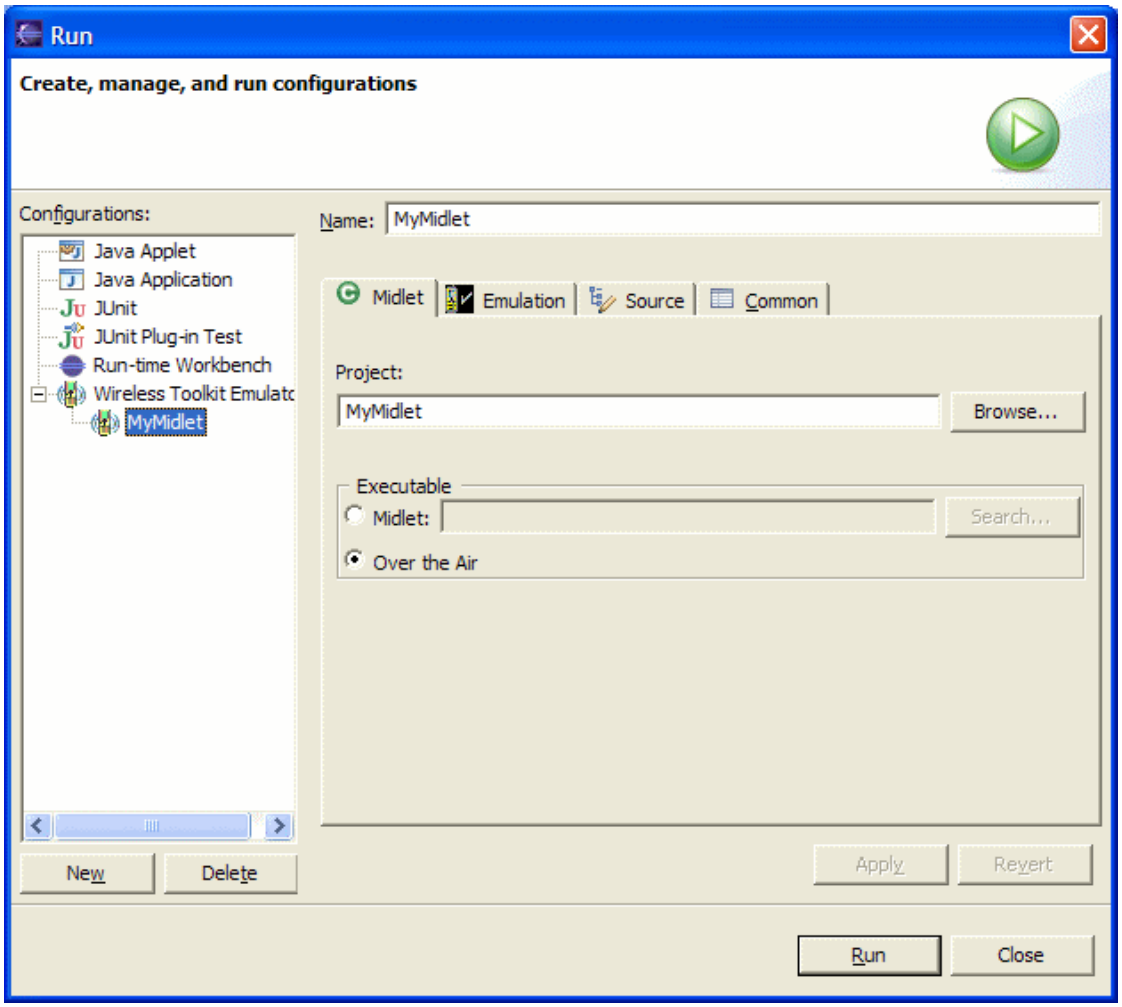
12. 参考

12.1. 启动设置

该对话框提供了对于无线工具模拟器的启动选项进行个别设置的一些细节：

- Midlet tab

MIDlet 面板控制着 Eclipse 将以什么样的方式来启动调试进程。你有两种选择：启动 MIDlet 或 者通过无线下载。



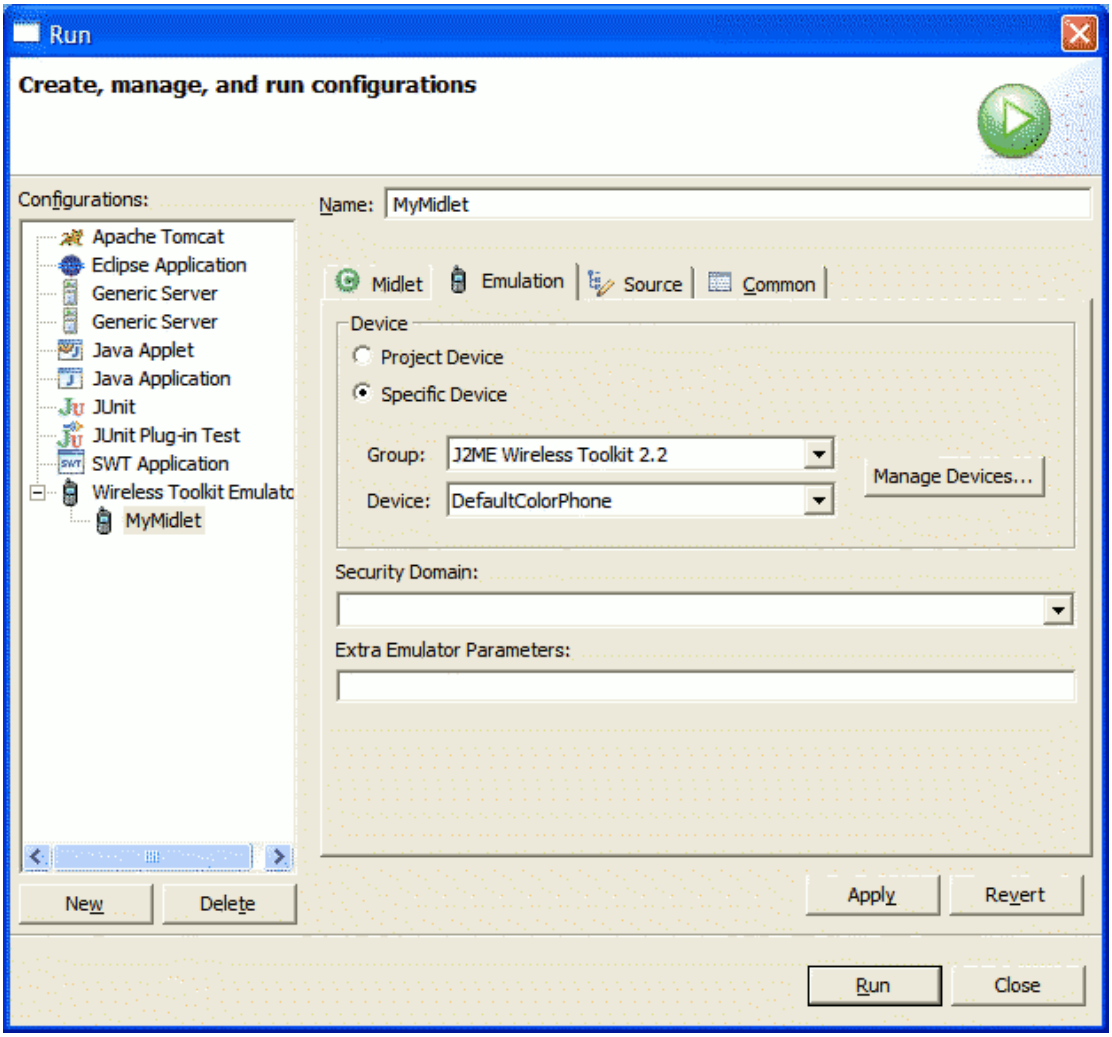
此面板上的项目是：

项目	内容
MIDlet Launching	当 MIlet launching（启动 MIDlet） 被选中，模拟器 将被告知从本地系统启动 MIDlet。 使用这种启动方法，在对 MIDlet 调试之前，你没有必要进行打包。当该选项被选中，你必须指定 MIDlet 的类名。
Over The Air	当 Over the Air（无线下载） 被选中，Eclipse 模拟器的 Java 应用管理 (Application Manager，JAM) 将会通知 EclipseME 用其内置的 HTTP 服务器来下

	载 MIDlet 套件。 从 0.5.5 版本开始, Over the Air 选项会默认在运行之前自动部署工程 。 如果你禁用了这些特性,那么在使用该模式之前，你必须手动的部署工程。

模拟器面板

面板可以控制 MIDlet 将会在哪个特定设备的模拟器上运行。那些特定设备的模拟器必须被配置过了，设定才会有效。



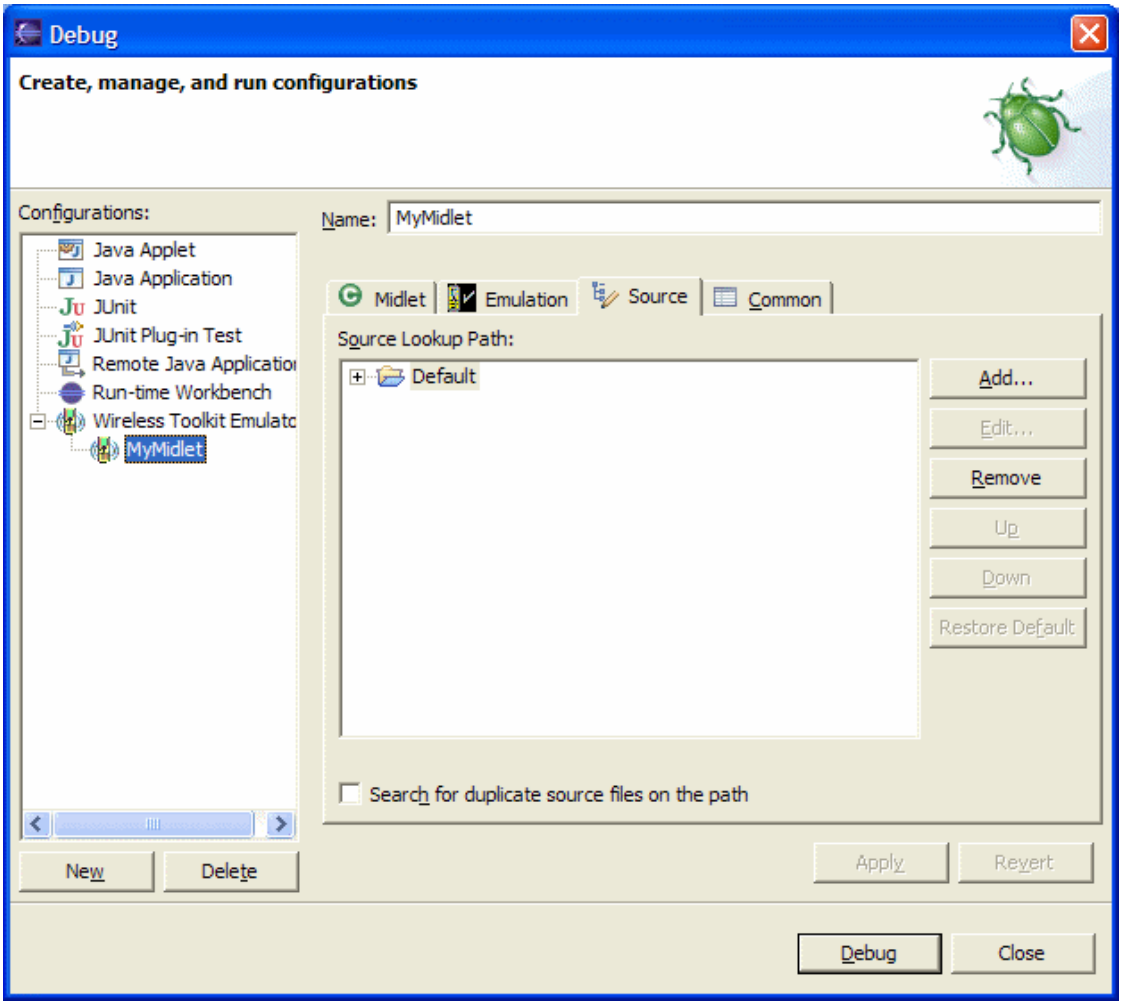
此面板上的项目是:

项目	内容
Project Device	该设定允许你选择 MIDlet 使用什么设备定义来模拟运行。 这是一个很有用的选项, 比如, 你为某个设备写了一个 MIDlet，现在你可以用另外一个设备定义来运行，测试一下效果如何 。 MIDlet 会默认的使用工程属性中定义的设备来运行。 要是你想使用另外一个设备的模拟器 的话，请选择单选框的第二个选项，并且从组合下拉框中选择你想要的设备。
Specific Device	该选项被选中的话，你可以指定一个你想用来运行你的 MIDlet 的模拟器
Security Domain	这个下拉列表框可以让你选择一个指定安全范围，并以此来运行 MIDlet。在调试一个被签署 过的 MIDlet 时，这个选项就显得很有用了。 详情可以参见关于签署 MIDlet 套件.

Extra Emulation Parameters	你可以在这里填上要直接传给模拟器的那些额外参数。这是很有用途的。比如说， 你可以打开 JVM/KVM 调试信息的选项。

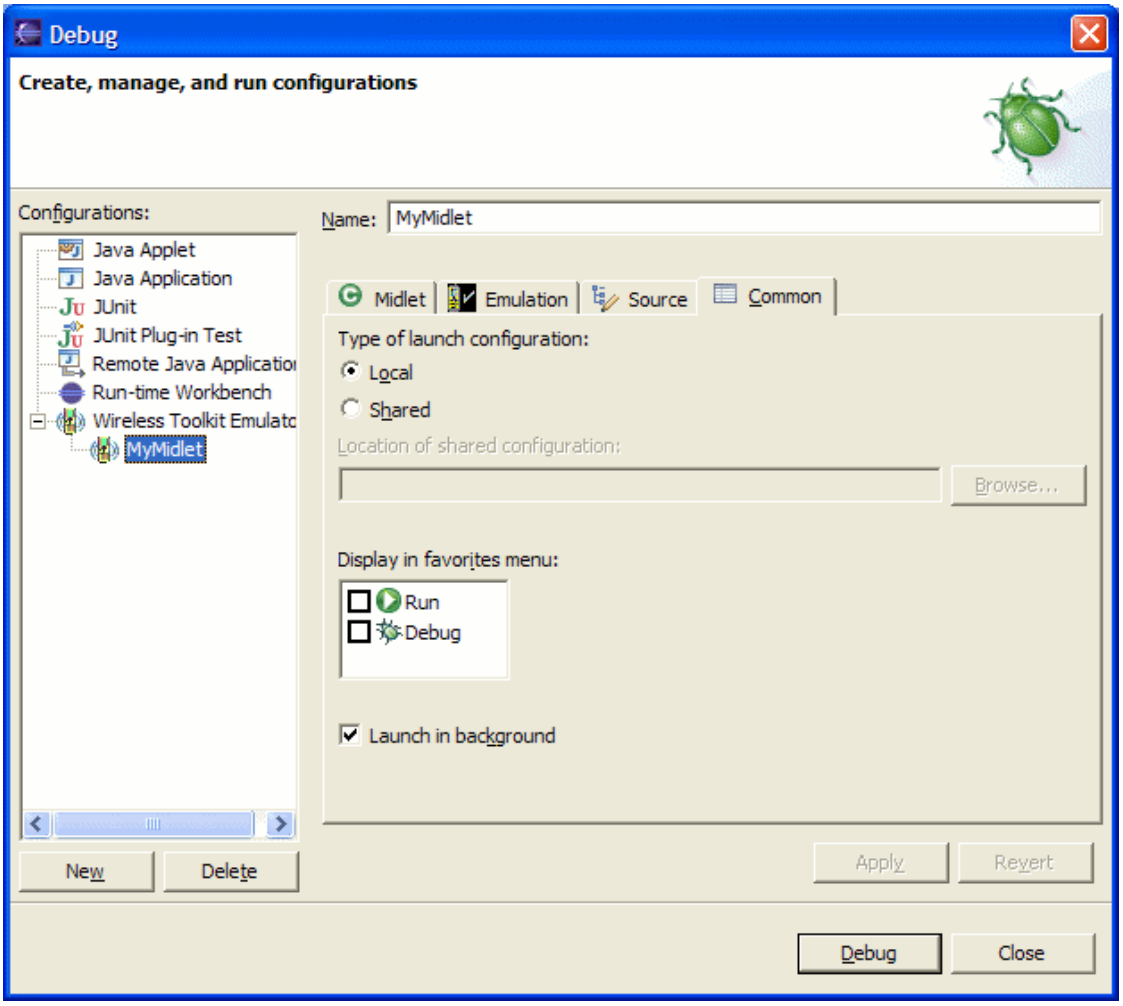
源代码面板

源代码面板是一个 Eclipse 的标准选项，你可以在这里配置你的源代码所在的目录。



通用面板

通用面板源代码面板是一个 Eclipse 的标准选项。在这里你可以选择你的启动配置是本地的(Local)还是共享的(Share)，以及它是否在运行和调试菜单中出现。

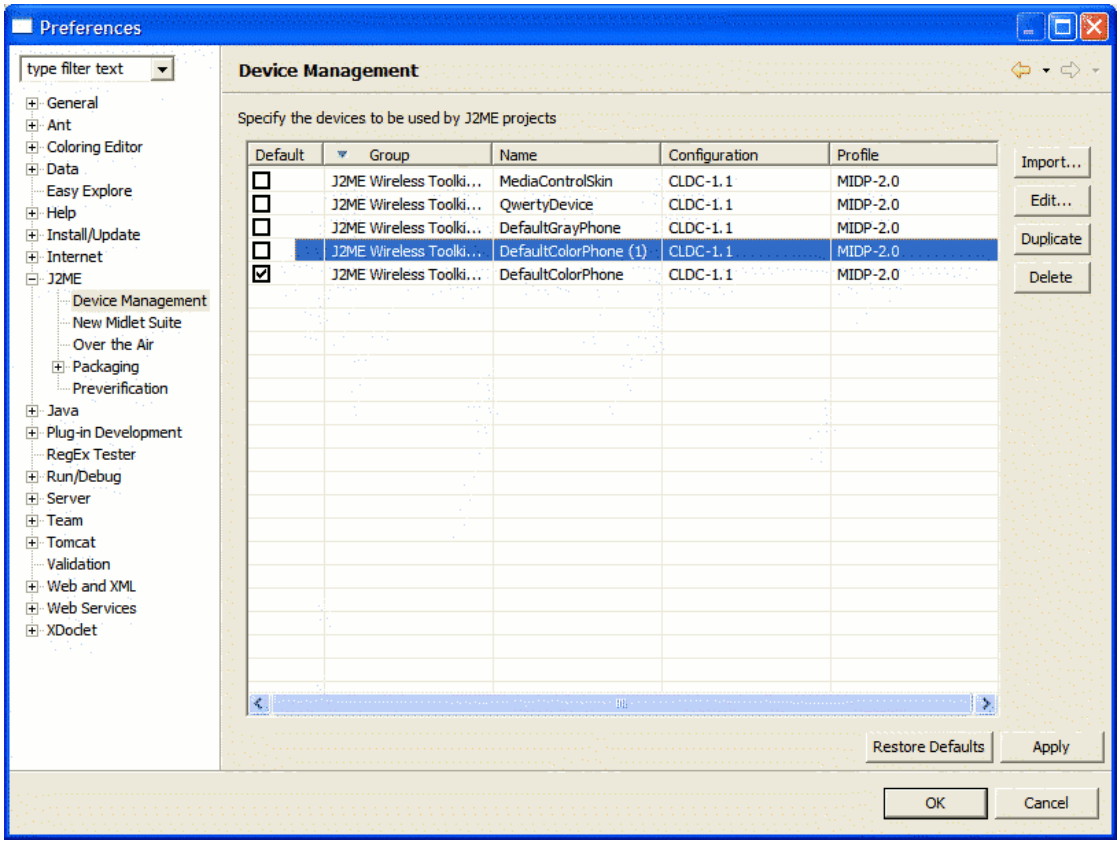


12.2. J2ME 首选项

为了查看或者调整 EclipseME 的首选项,你可以从 Window 菜单打开 Preferences 对话框。本文档描述了接下来的一系列设置。

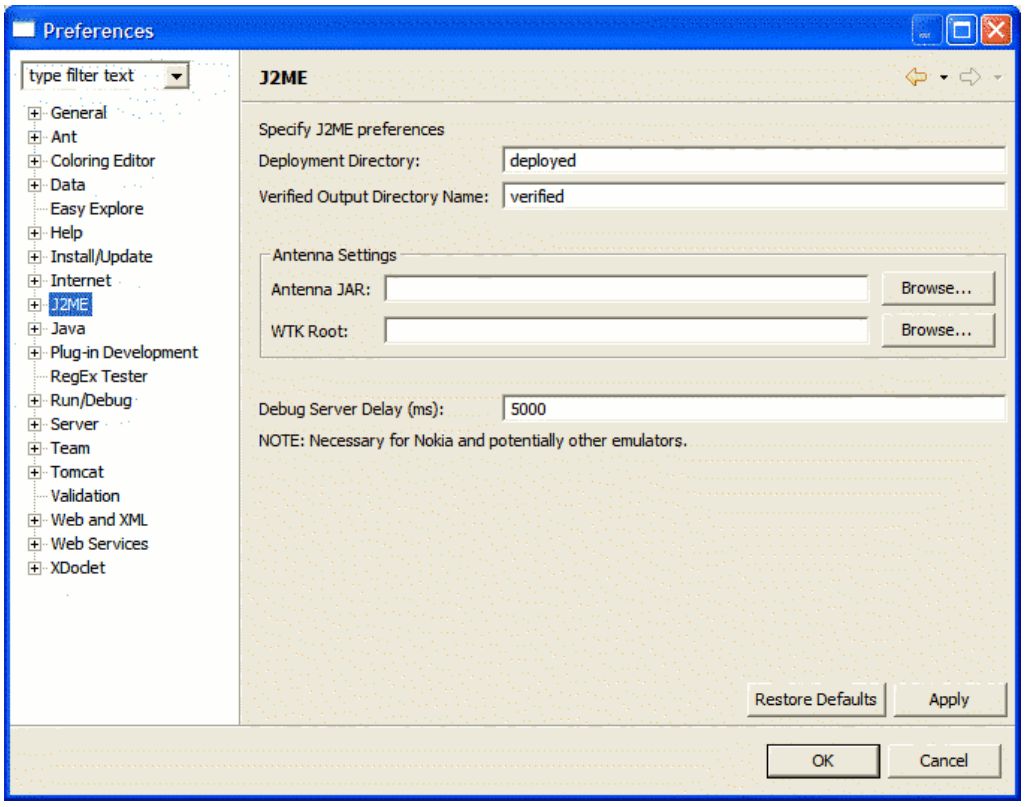
- 设备管理

你可以从首选项面板左边目录选择 J2ME/Device Management 来设定一些首选项，用来控制被安装到 EclipseME 的设备定义。 关于设备定义的详情，请参见设备管理一节所描述的信息。



基本首选项

从首选项面板左边目录选择 J2ME 栏目后，你就会看到一个基本首选项设置 页,如下图所示:

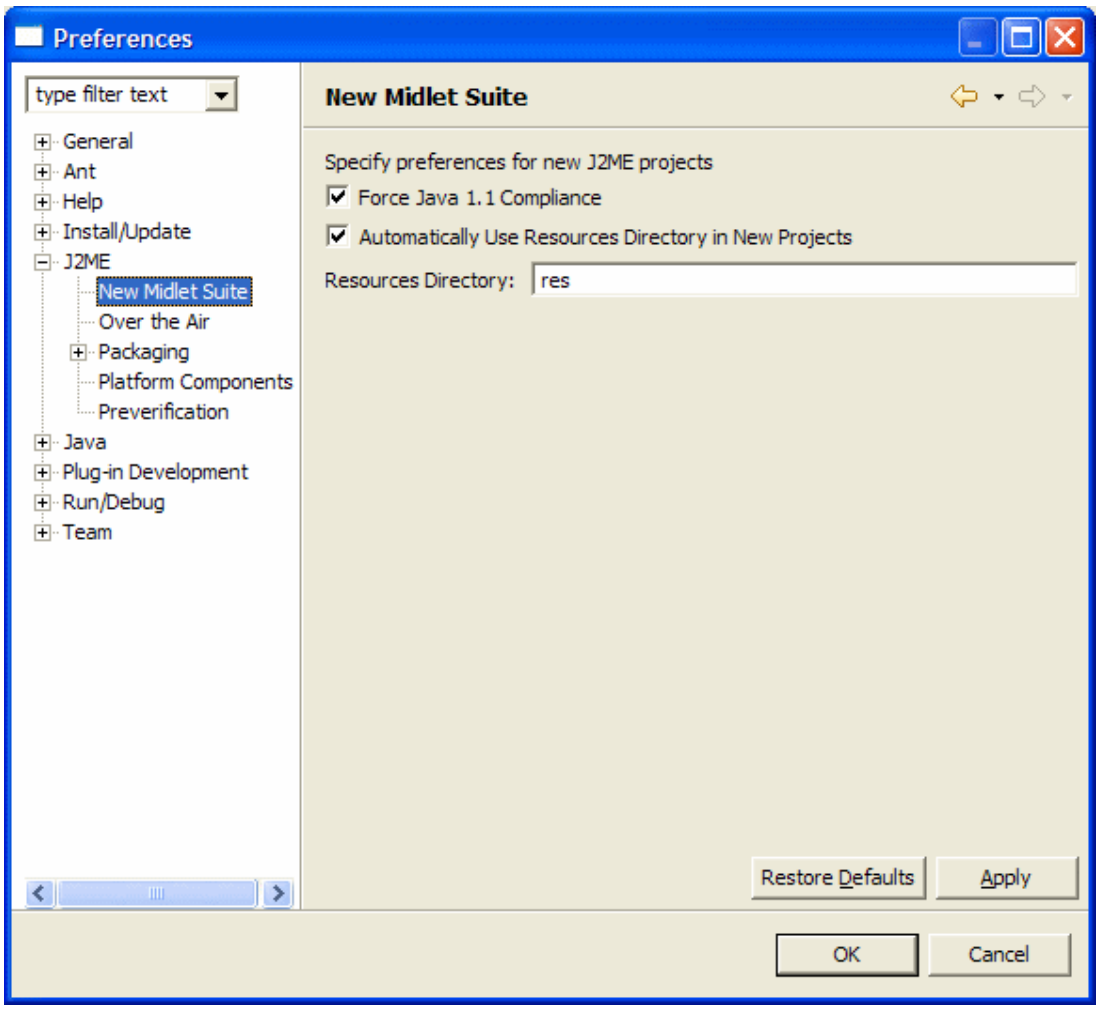


首选项项目:

项目	内容
Deployment Directory	Deployment Directory（部署目录）选项允许你设置一个相对于工程目录的子目录， 当你对工程执行打包操作时所生成的 JAR 以及 JAD 文件会被 到该目录中。另外，内置的 无线下载 部署功能也会从这个目录下去寻找 JAD 以及 JAR 文件。
Verified Output Directory Name	J2ME 规范规定所有的类在运行前都必须经过预校验。 这个规定带来了较高的安全机制而又不像 J2SE 那样需要很大的开销。 如果 Eclipse 工作环境运行自动编译，那么 EclipseME 可以对类文件自动进行预校验。 当 Eclipse 对一个类完成编译以后，EclipseME 将会对其进行预校验并且把它放到该选项设置的目录中去。
Antenna JAR	如果你想使用 EclipseME 自带的生成 Antenna 构建文件的功能的话,那么 你必须告诉 EclipseME 你把 Antenna 的 JAR 文件放到哪儿了。详情可以参见 Antenna 支持一节。
WTK ROOT	如果你想使用 EclipseME 自带的生成 Antenna 构建文件的功能的话,那么 你需要让 EclipseME 知道 WTK 的安装目录在什么地方。详情可以参见 Antenna 支持一节。
Debug Server Delay	有些模拟器，比如说 NOKIA 的，调试的时候是以服务端而不是客户端的形式出现， 所以这类模拟器必须在 EclipseME 试图连接它之前启动。由于有很多因素会影响 启动的时间，为了让模拟器有足够的时间启动，所以在这里你可以设定在等待 多少时间（毫秒）以后，EclipseME 才去连接模拟器的调制服务端。

新建 Midlet 套件首选项

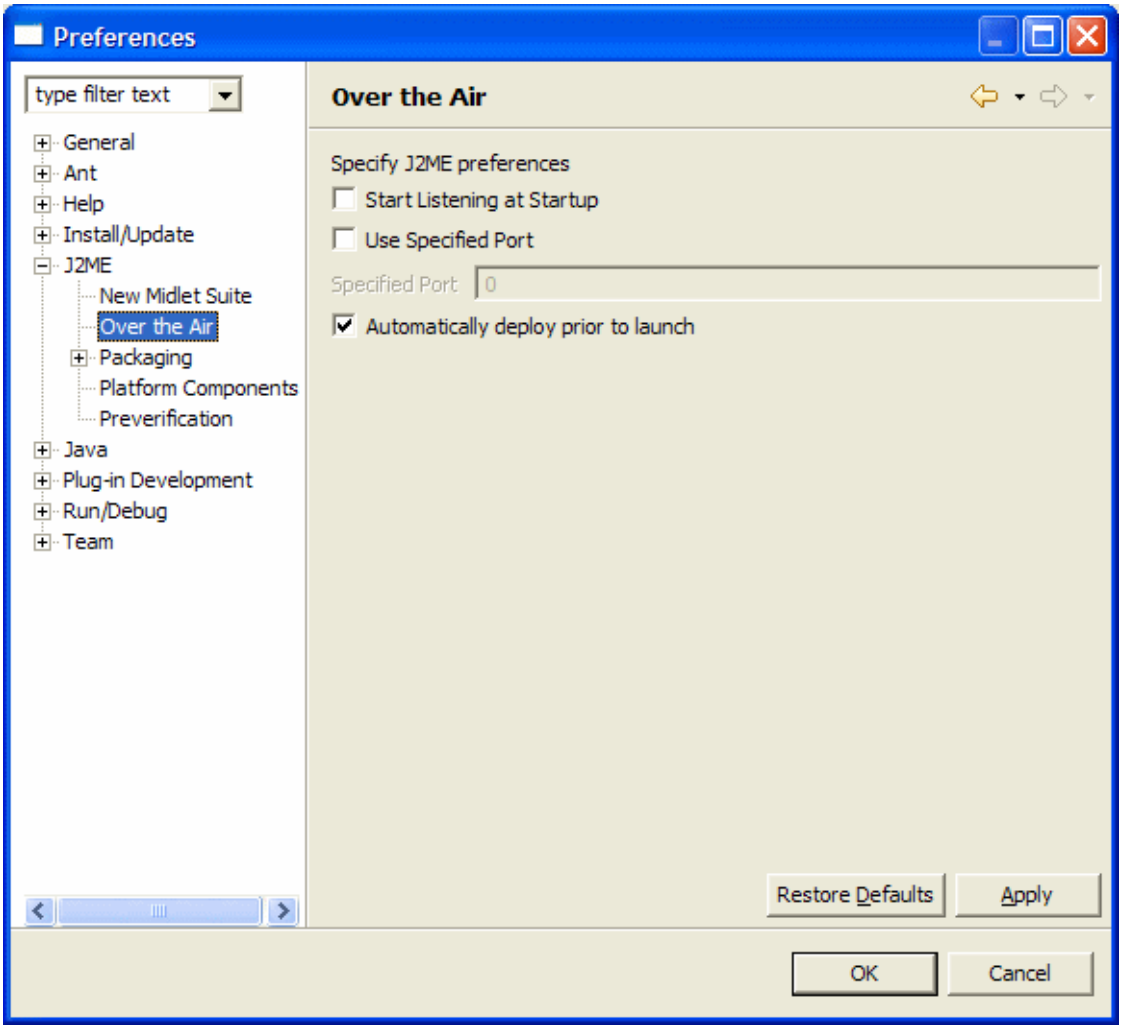
从首选项面板左边目录选择 New Midlet Suite 目录，你可以设定一些关于 新建一个 Midlet 套件的首选项。



条目	内容
Force Java 1.1 Compliance	如果该项被选中了，EclipseME 创建工程文件的时候会自动调整工程属性的设定， 使编译器强制生成 Java 1.1 版本的类文件。如果没有选中的话，那么就使用工作环境的设定。 警告：因为很多手持设备都不支持 1.1 以后版本的类文件格式，所以我们强烈推荐你选中这个选项。
Automatically Use Resources Directory in New Projects	如果该选项被选中，那么当你创建 MIDlet 工程的时候，同时会自动创建一个资源文件目录， 该目录的文件名由下面那个文本输入框的内容决定。
Resources Directory	如果上面那个选项被选中的话，那么你可以在这里设置一个资源文件夹的名字。注意这个特性 只有在新建工程时管用，并且你要设置了让工程的源文件和二进制文件使用不同的目录。

无线下载首选项

从首选项面板左边目录选择 J2ME/Over the Air 后，你可以设定 EclipseME 的无线下载首选项。

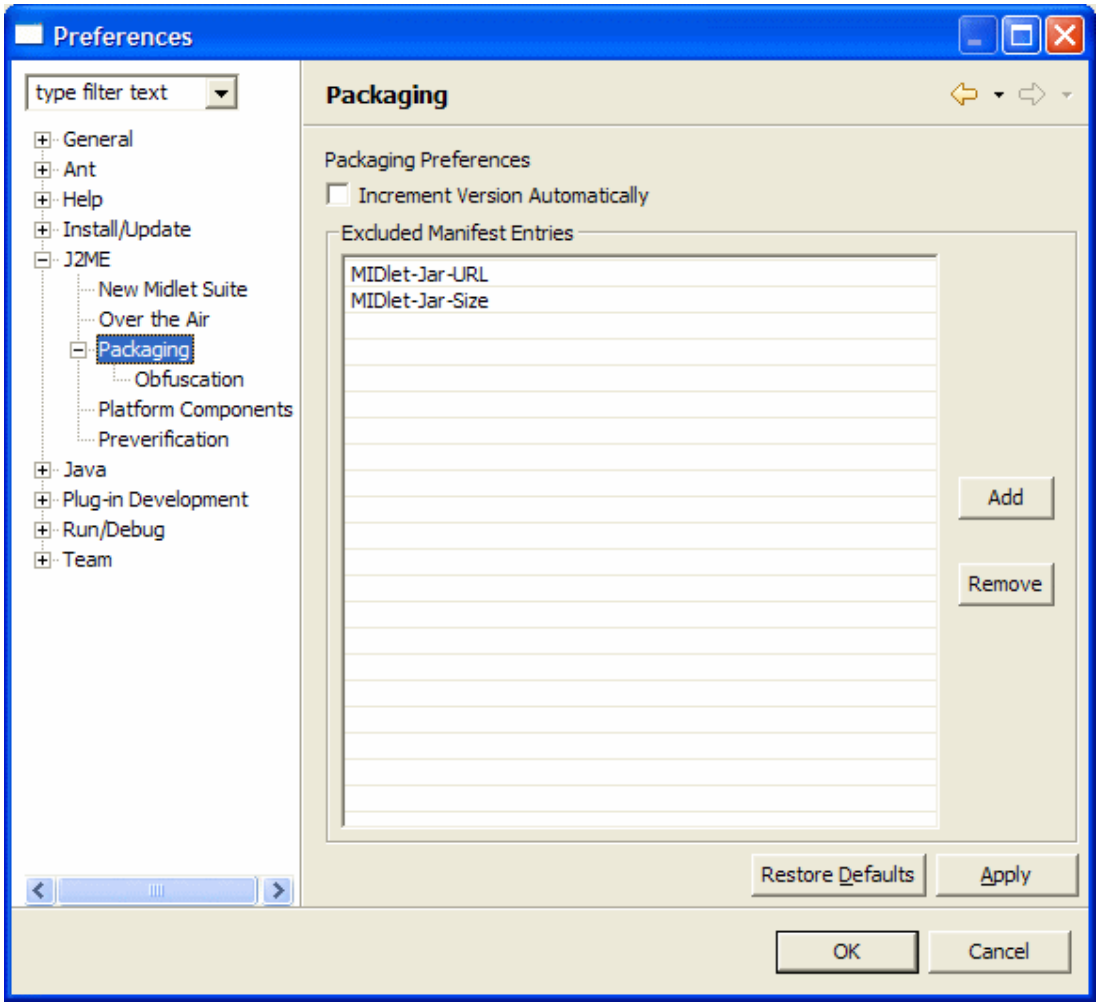


首选项项目:

项目	内容
Start Listening at Startup	如果该选项被选中，那么当 Eclipse 环境打开的时候会启动一个内置的无线下载服务器。要是在 Eclipse 环境以外使用 OTA（无线下载）服务器 的话，请确保该服务器已经被启动了并且端口指定正确。如果可以在 Eclipse 里启动模拟器，那么你没有必要去启动服务程序了，因为必要的时候它会自动 启动的。
Use Specified Port	选中该选项并且填入一个端口可以为内置的 HTTP 服务程序指定一个端口号。 如果该选项没有被选中的话，那么 OTA 服务程序会被随机的指定一个端口。
指定端口	当 Use Specified Port 选项被选中， 为内嵌的 HTTP 服务程序指定一个端口。
Automatically deploy prior to launch	为了执行无线下载运行操作，JAR 和 JAD 必须预先部署完毕。选中该选项后，如果自从 上次打包后程序又有了变动，EclipseME 会自动在执行无线下载操作之前打包你 的 MIDlet 套件。

打包首选项

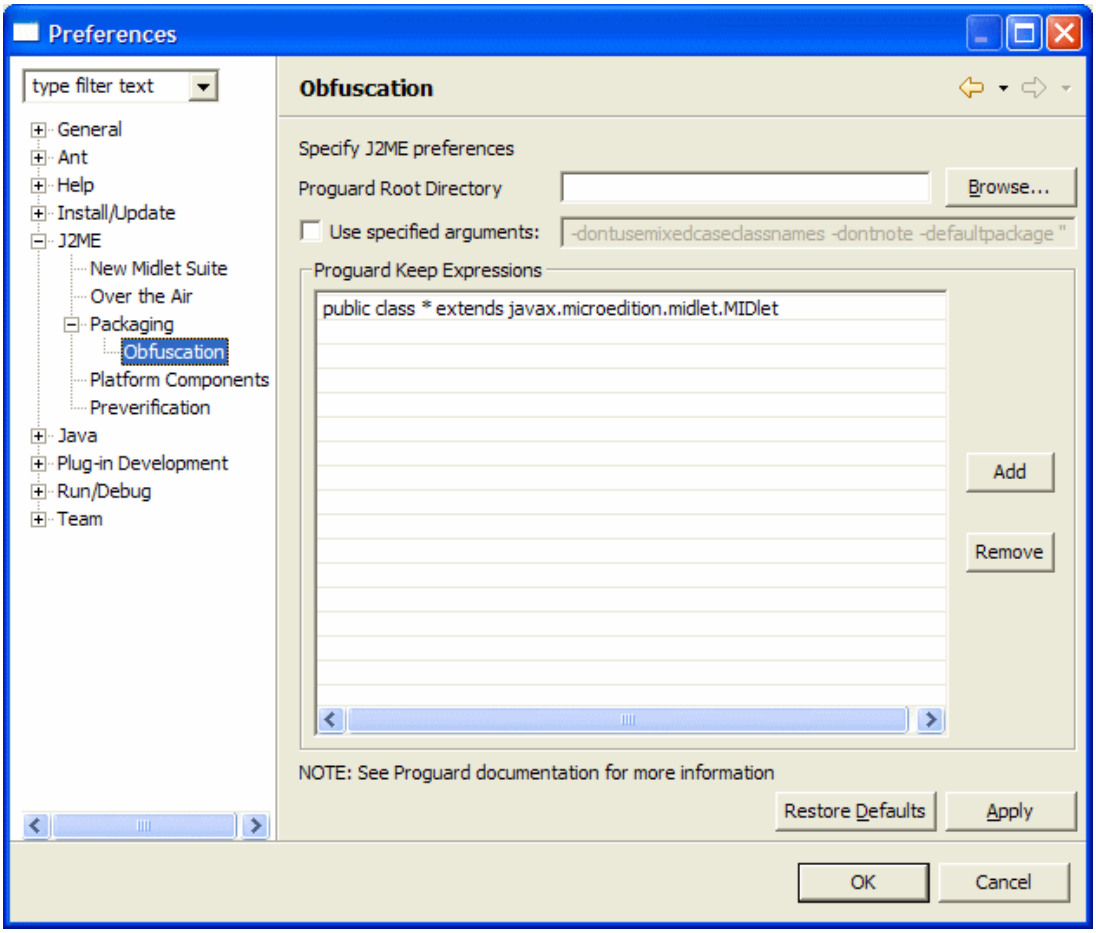
打包首选项可以对将要发布的文件进行打包的过程进行控制。



项目	内容
Increment Version Automatically	如果该选项被选中，每次 MIDlet 套件重新打包后，EcilpseME 会自动的增加 Midlet Version 参数的值。 注意：从 EclipseME 1.2.0 开始，如果一个版本号的形式为 x.y.z，当 y 或者 z 的数值增加到 99 了，就将该位回滚到 0，并且将前一位加 1。这么做是因为如果 JAD 文件中的版本号含有三位数的部分会引起手机的错误。
排除属性	在这里，你可以设置一些 MIDlet 属性，使得打包的时候它们不会出现在 JAR 文件中的 manifest 文件里。默认情况下，MIDlet-Jar-URL 和 MIDlet-Jar-Size 会被排除。

混淆首选项

混淆首选项控制对包进行混淆的过程。 EclipseME 使用 Proguard 来完成混淆过程。



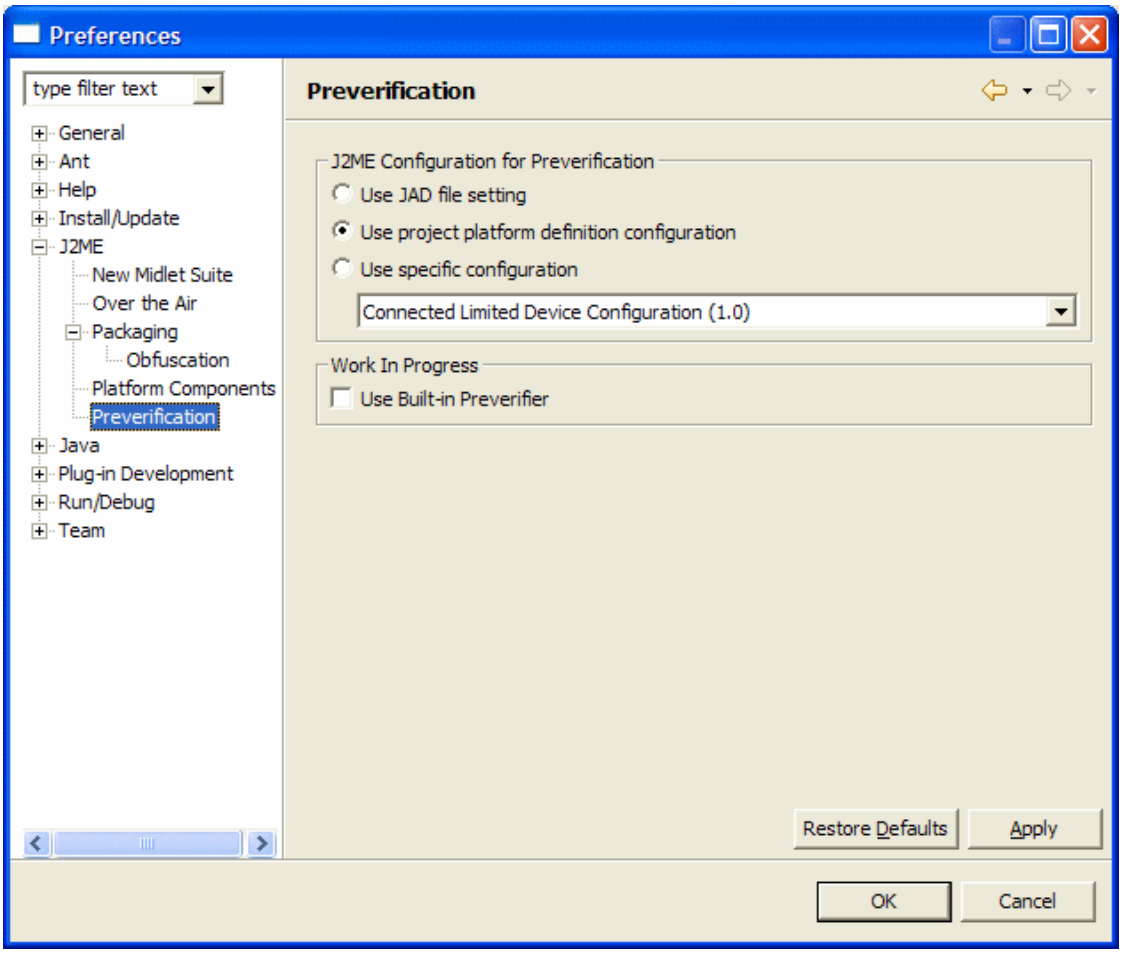
首选项项目:

项目	内容
Proguard Root Directory	在这里你可以指定 Proguard 安装路径。 如果你要对你的程序混淆，必须安装 Proguard。
Use Specified Arguments	默认情况下，EclipseME 会使用如下参数来调用 Proguard 进行混淆： -dontusemixedcaseclassnames -dontnote -defaultpackage '' 在某些场合你需要指定不同的参数时，可以将"Use specified arguments"（使用指定参数） 这个选项选中并且在文本框里填上参数。 Microsoft Windows 用户特别注意：默认情况下，ProGuard 假定你使用的操作系统会区分文件名中的大小写。(比如 A.java 和 a.java 是两个不同的文件)很明显，Microsoft Windows 不是这样的操作系统。 Windows 用户要确定为 ProGuard 指定了- dontusemixedcaseclassnames 选项。 如果你没有这样做，而你的工程文件又超过了 26 个类的话，ProGuard 默认 使用变换大小写的方式为混淆后的类命名。显然这会覆盖原来的类。 安全起见，从 EclipseME 0.9.0 发行版开始 -dontusemixedcaseclassnames 选项被包含到了传给 ProGuard 的默认选项中去。 如果你是*NIX 用户并且你的工程中的类超过 26 个的话， 去掉该选项你可以得到一个稍微小点的 JAR 文件。

Proguard Keep Expressions	在这里你可以设置一个表达式是的某些类不被混淆。表达式中用到的关键字请参阅 Proguard 文档。 默认的表达式会保留所有的 MIElet 类以及他们的方法不被混淆。

预校验首选项

从首选项面板左边目录选择 J2ME/Preverification 目录 你可以对该插件的预校验操作进行设置。



这里主要有两个关键问题需要设置:

为 CLDC 1.0 进行预审和还是 CLDC 1.1。

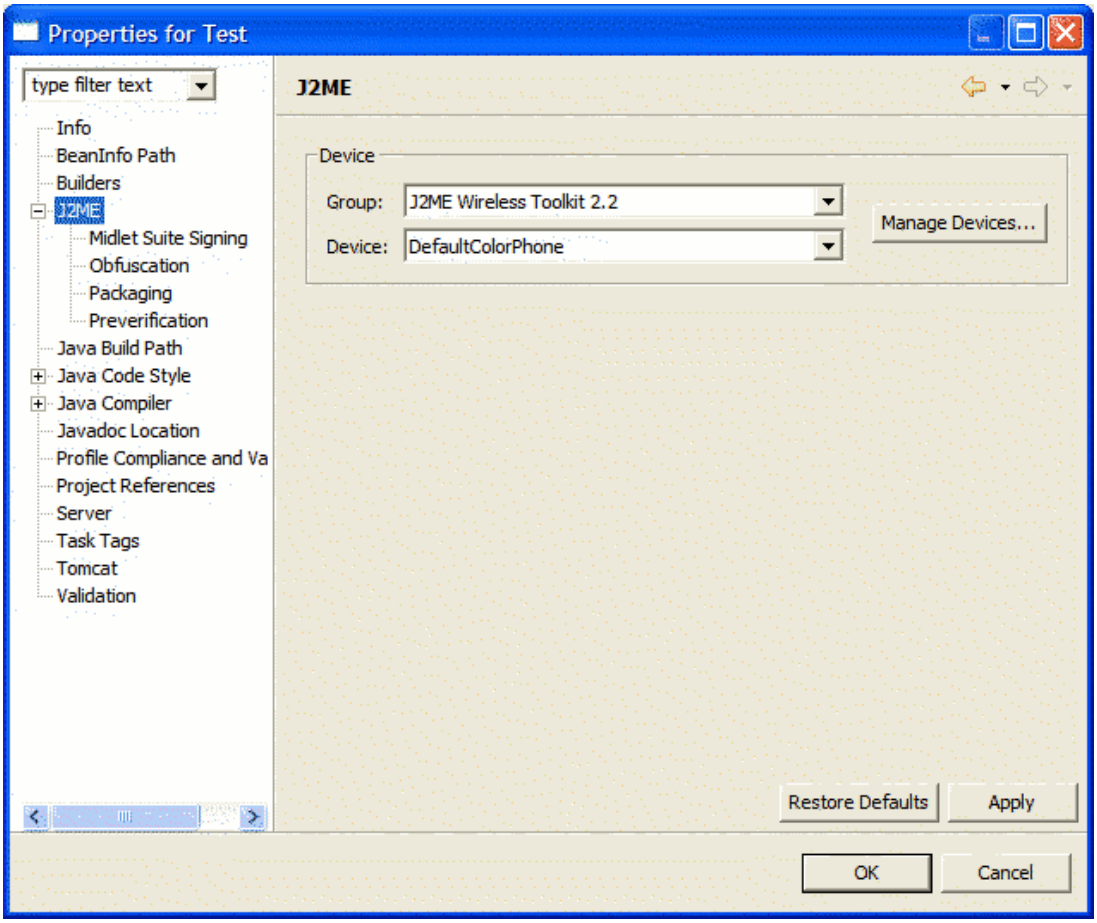
是使用外部 WTK 的预校验器还是使用内部的预校验器

项目	内容
Use JAD file settings	该选项被选中的话，EclipseME 会根据 JAD 文件中的配置来自行决定是使用 CLDC 1.0 还是 CLDC 1.1 的校验方式。
Use project platform definition configuration	如果这个选项被选中的话，EclipseME 会根据工程平台定义的配置情况来决定是使用 CLDC 1.0 还是 CLDC 1.1 的形式。
Use specific configuration	这个选项允许你从下拉列表里面指定 CLDC1.0 或者 1.1,而无论 JAD 中定义如何或者 使用什么样的工程平台定义。
Use Built-In Preverifier	自从 EclipseME 1.2 版本开始, EclipseME 包含了一个预校验器的本地 JAVA 实现。如果 你选中这个选项的话，那么 EclipseME

	会用它来替代外部 WTK 的预校验器。 注意：内部的预校验器还在还处在 beta 阶段，不保证一定好使哦。 不过内部预校验器遇到预校验错误时的反馈功能还是不错的。 EclipseME 开发者会对该预校验器的反馈功能很满意的。

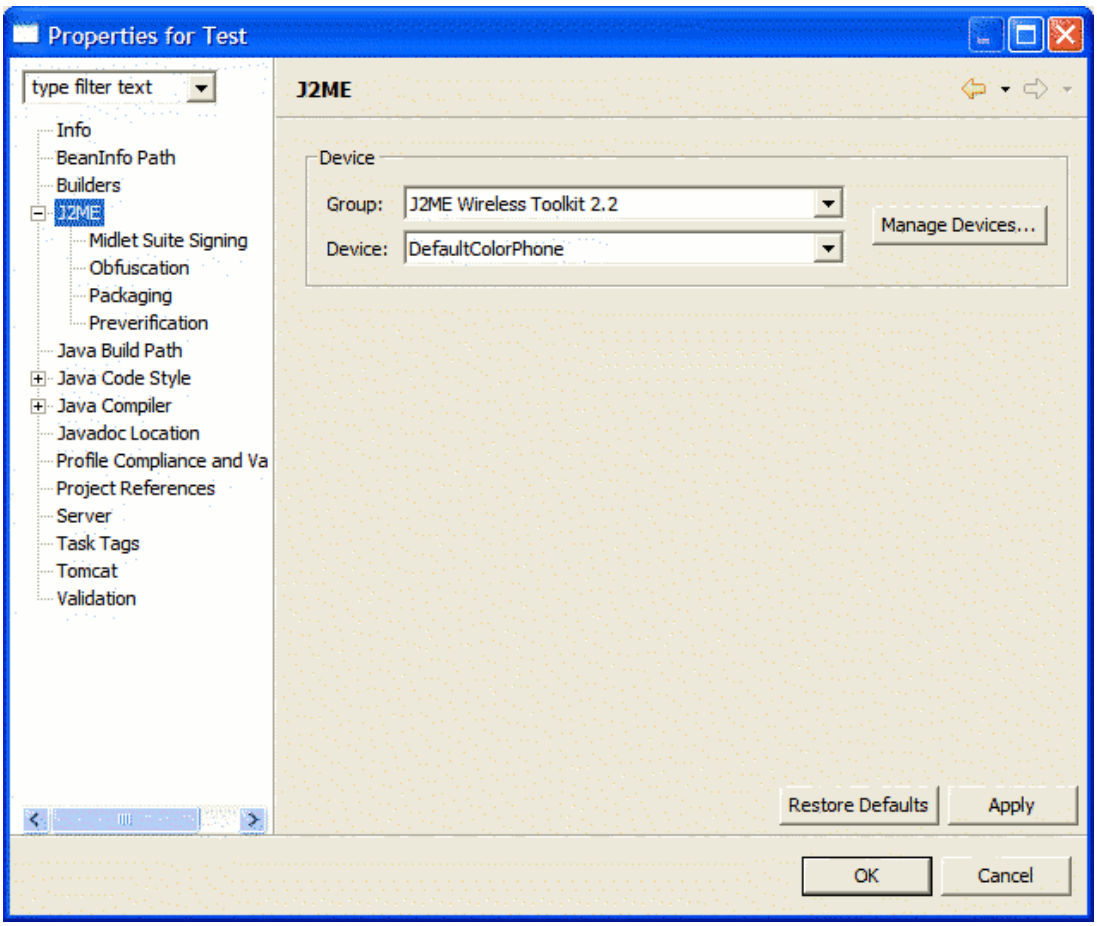
12.3. 项目属性

选择 Eclipse 环境的 Project/Properties 菜单打开属性对话框，你就可以查看或者修改 EclipseME 项目的属性。



J2ME 页

从属性对话框的左侧面板选择 J2ME 目录，你会看到如图所示的属性页：

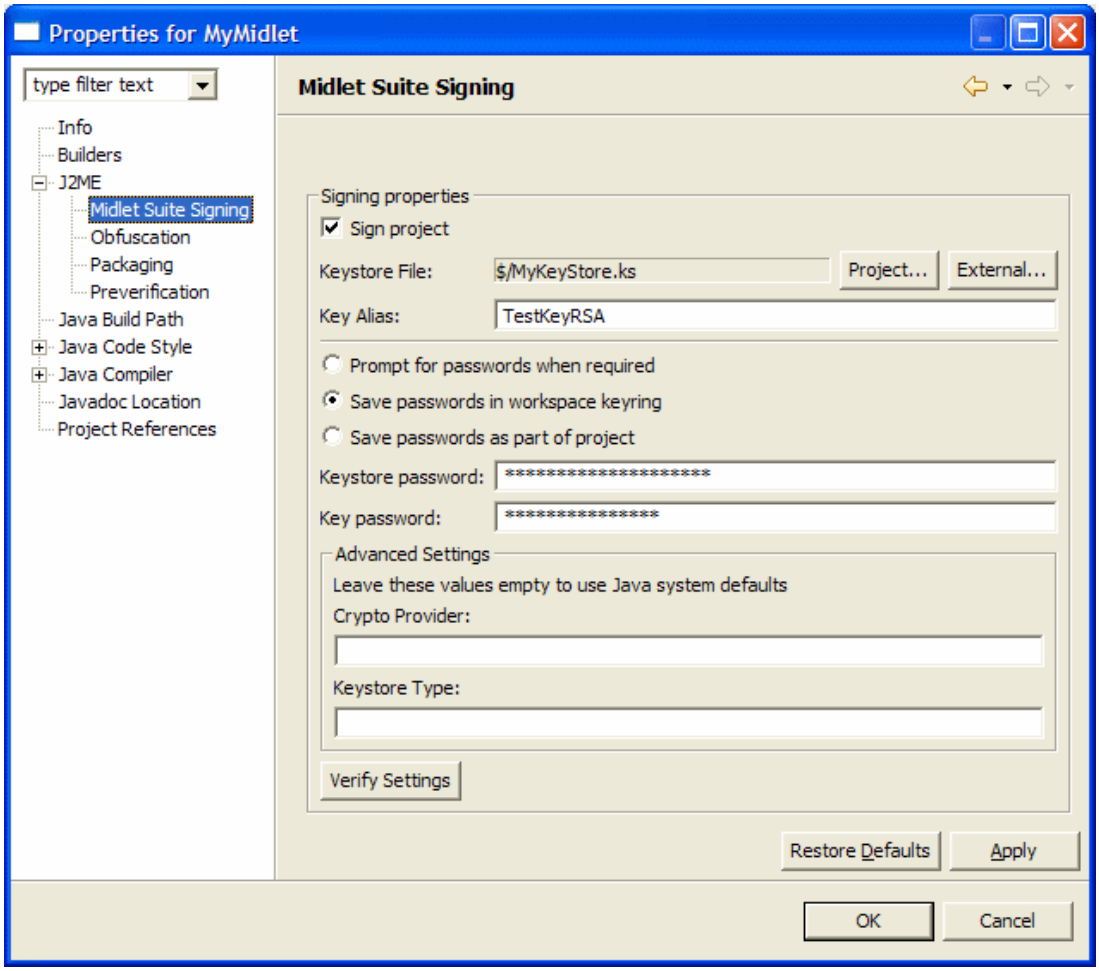


属性项目:

Item	内容
Device	该设置不但决定了使用什么样的特定设备来编译工程，还决定了运行该项目时的默认设备。 当该设定被修改的话，整个工程会自动再编译一遍。 所以当你的项目中使用了一些 MIDP2.0 独有的类或者方法时候，如果你把该属性从 MIDP2.0 平台修改成 MIDP1.0 平台的话，会通不过编译。

签署 MIDlet 套件页

从属性对话框的左侧面板选择 MIDlet Suite Signing 子菜单 会显示关于签署项目的页面，如下图所示：



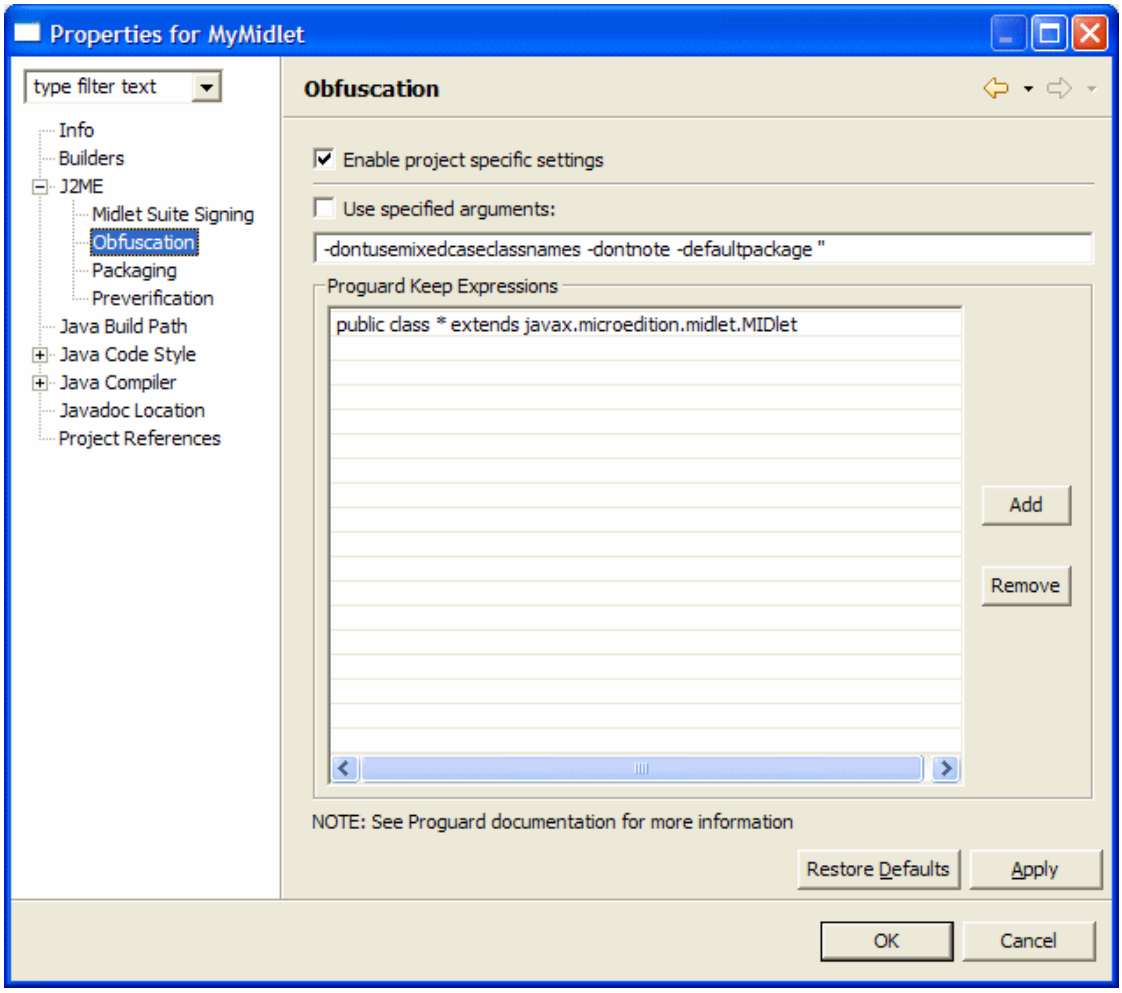
属性项目:

Sign project	选中该选项会使签署项目的功能生效。注意，如果你选中该选项的话，你必须至少有一个 密钥库文件(KeyStore)以及匹配的密钥别名(Key Alias)。
Keystore File	该设置用来指定一个密钥库文件。这个文件包含了用来签署 MIDlet 套件和认证签署有效性的的密钥。
Key Alias	该选项指定了所使用密钥的别名。
Prompt for password when required	当该选项被选中的话，EclipseME 会在从密钥库里提取信息的时候提示你输入 密钥库(Keystore)和密钥的密码。 注意：EclipseME 会把你输入的密码记录在内存里，所以在该 Eclipse 进程里你不要 在此输入密码。
Save passwords in workspace keyring	如果你选择了该选项并且在文本框内输入了密钥库(keystore)和密钥(key)的密码，EclipseME 会将这些 密码保存到你的 keyring 文件， 默认情况下，该文件位于［工作 目录］\.metadata\.plugins\org.eclipse.pde.core\eASEE\org.eclipse.core.runtime\.keyring， 当然你也可以使用 Eclipse 的-keyring 指定到另外一个文件。

Save passwords as part of project	如果你选择该选项并且在文本框内输入了 keystore 和 key 的密码，EclipseME 会将这些密码 保存在.eclipseme 文件中，使他们成为项目的一部分。
Crypto Provider	如果你使用了非标准的密码术提供商，比如 Bouncy Castle 密码算法库，那么在这里输入提供商的 ID。如果你使用的是标准的系统提供商， 请把这里留空。
Keystore Type	如果你使用了非标准的密钥库格式，比如 Bouncy Castle 支持的特殊格式， 那么在这里输入密钥库格式的 ID。如果你使用的是标准的系统密钥库类型，请把这里留空。
Verify Settings	点击这个按钮会导致 EclipseME 尝试从密钥库中获取密钥和证书，并检查以确保此密钥的类型正确。 如果此测试成功了，那么这个密钥就可用于对 MIDlet 套件进行签名。 如果你输入的参数有一处或更多处不正确，你会得到相应的错误消息提示。一般来说我们建议在签名部署项目之前应对配置进行验证。

混淆页

从属性对话框的左侧面板选择 Obfuscation 子目录你会看到混淆页。它看起来如图所示：



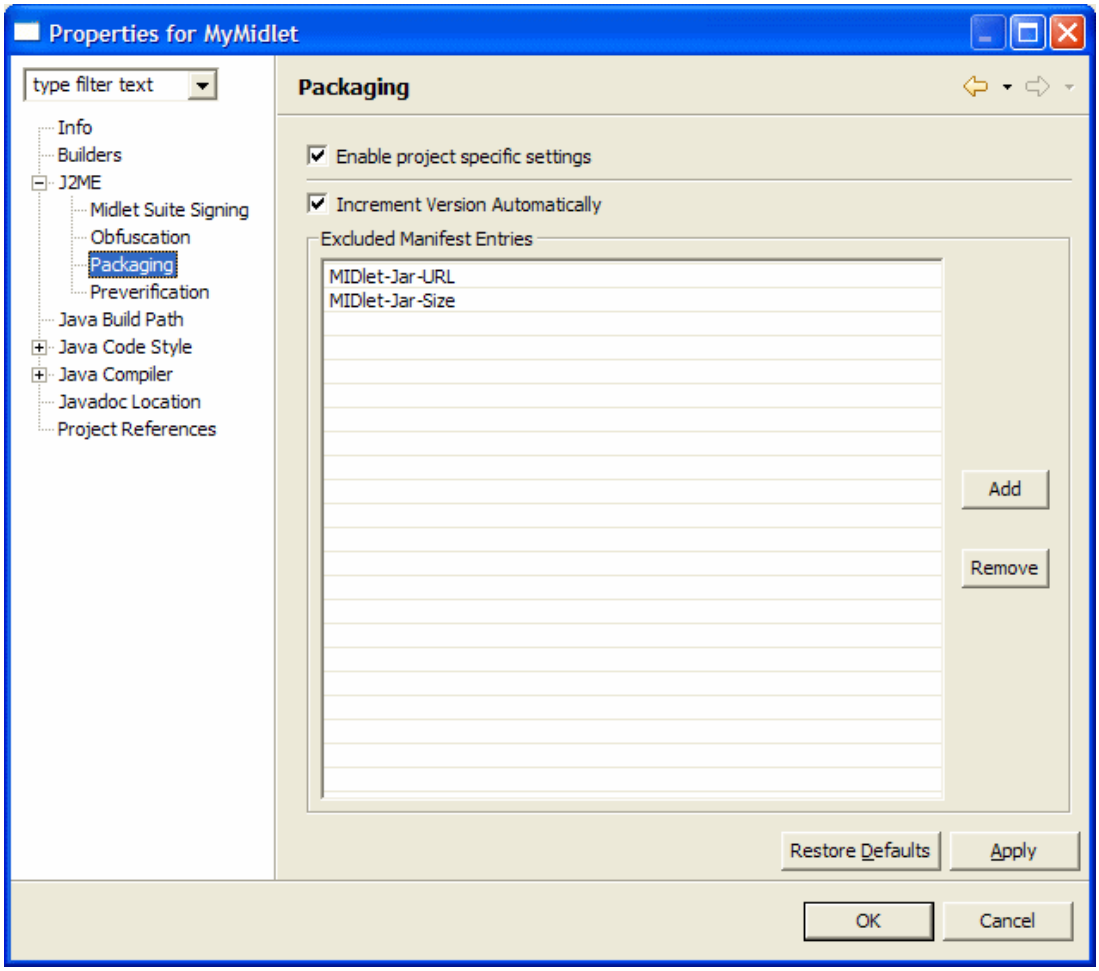
属性项目:

项目	内容
Enable project specific settings	如果该属性未被选中，那么 Eclipse 会使用 混淆首选项中所指定的全局配置。 如果被选中的话，本页的设置将会替代全局设置。

参见混淆首选项一节的描述以得到更多信息。

打包页

从属性对话框的左侧面板选择 Packaging 子目录你会看到 打包属性页。它看起来如图所示:



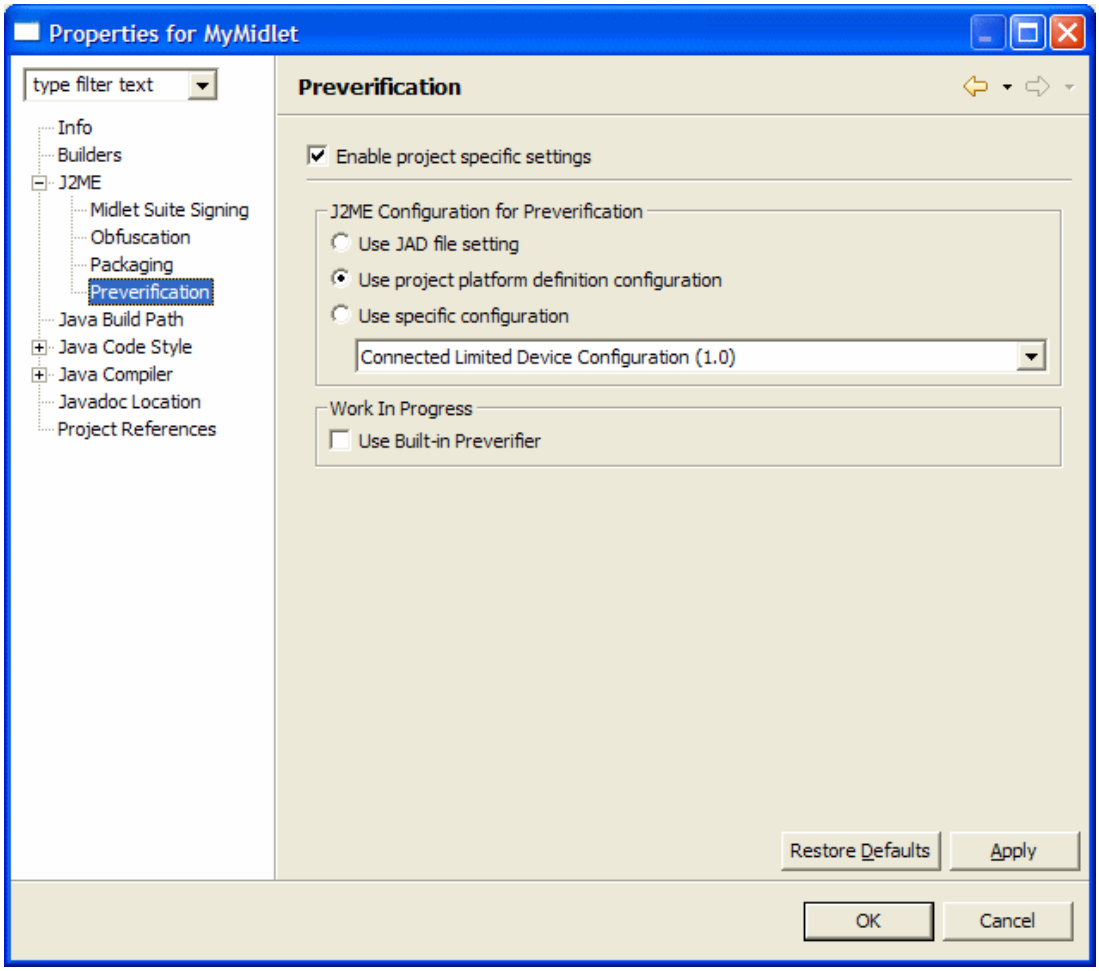
属性项目:

项目	内容
Enable project specific settings	如果该属性未被选中，那么 Eclipse 会使用打包首选项中所指定的全局配置。 如果被选中的话，本页的设置将会替代全局设置。

参见打包首选项一节的描述以得到更多信息。

预校验页

从属性对话框的左侧面板选择 Preverification 子目录你会看到 打包属性页。它看起来如图所示:



这里主要有两个关键问题需要设置:
为 CLDC 1.0 进行预审和还是 CLDC 1.1。
是使用外部 WTK 的预校验器还是使用内部的预校验器
属性项目:

Item	Contents
Enable project specific settings	如果该属性未被选中，那么 Eclipse 会使用预预校验选项
Use JAD file settings	该选项被选中的话，EclipseME 会根据 JAD 文件中的配置来自行决定是使用 CLDC 1.0 还是 CLDC 1.1 的校验方式。
Use project platform definition configuration	如果这个选项被选中的话，EclipseME 会根据工程平台定义的配置情况来决定是使用 CLDC 1.0 还是 CLDC 1.1 的形式。
Use specific configuration	这个选项允许你从下拉列表里面指定 CLDC1.0 或者 1.1,而无论 JAD 中定义如何或者 使用什么样的工程平台定义。
Use built-in preverifier	自从 EclipseME 1.2 版本开始, EclipseME 包含了一个预校验器的本地 JAVA 实现。如果 你选中这个选项的话，那么 EclipseME 会用它来替代外部 WTK 的预校验器。 注意: 内部的预校验器还在还处在 beta 阶段，不保证一定好使哦。 不过内部预校验器遇到预校验错误时的反馈功能还是不错的。

	EclipseME 开发者会对该预校验器的反馈功能很满意的。

12.4. Antenna 支持

- Antenna 概述

Antenna (http://antenna.sourceforge.net) 是Ant 的一个扩展项目。Antenna 提供了一套方便开发基于MIDP的无线JAVA程序的Ant任务。使用 Antenna,你可以进行编译 ，预校验，打包，混淆，运行你的MIDP程序以及生成JAD文件。另外它还可以把JAR文件转换 成PRC格式，这是一种由Sun和IBM实现的可以在PalmOS上运行的MIDP格式。所以我们可以看到 在Ant世界中Antenna可以完成很多工作，就好像EclipseME在IDE世界中一样。

从 0.6.0 版本开始, EclipseME 就可以自动生成和更新 Antenna 构建文件。这使得 EclipseME 用户可以轻松的将 Antenna 构建文件发布给非 Eclipse 用户。或者将 Antenna 与 EclipseME 结合完成一些操作，比如将 PRC 格式的转换，这个功能 EclipseME 是不提供的。 另外这也使得 EclipseME 项目可以包含在一个大项目中进行自动构建。

- 配置

在你使用 EclipseME 的自动生成 Antenna 构建文件的功能之前, 你必须首先告诉 Eclipse Antenna 的 Jar 文件在 什么地方以及 WTK 的安装目录在什么地方。 该配置项是基本首选项的一部分。

- 生成 Antenna 构建文件

为了准确的导出 Antenna 构建文件，你所需要做的是在 Package Explorer 窗口对工程点 右键，然后选择 J2ME 选项的 Export Antenna Build Files 子项。

第一次完成该操作后，你的工程里被添加进来三个新文件:

build.xml

这是标准 Ant 构建文件。 如果你的工程原来没有 build.xml 文件， 那么会新加一个进来。

eclipseme-build.xml

这是所生成的 Antenna 构建文件。

eclipseme-build.properties

该文件含有 eclipseme-build.xml 文件需要导入的属性。

Eclipse 内置了对 Ant 的支持,而 EclipseME 内置了对 Antenna 的支持,所以 为了运行 Antenna 构建文件你只需要直接在 build.xml 点击右键后选择 Run / Ant Build 或者从主菜单的 Run 中选择 External Tools

- 自定义 Antenna 构建

尽管自动生成的 build.xml and eclipseme-build.xml 文件包含了大多数你会用到的 antenna 典型操作,不过 EclipseME 的设计者显然是 不可能顾及到所有的情况的。所以 EclipseME 提供了自定义构建文件的能力。

不要修改 eclipseme-build.xml 和 eclipseme-build.properties 文件。

当你再次执行导出操作时,这些文件会被重新生成,你对这些文件的所有修改都会被覆盖。 此外 eclipseme-build.xml 的组织和形式还经常会改变。

如果你想为自动生成的 eclipseme-build.properties 文件增加属性，或者是修改属性，你可以创建一个名为 user-build.properties 的文件。文件中的内容会覆盖 EclipseME 自动生成

文件 的默认值。
如果你想为你的构建增加任务，那么就增加到 build.xml 文 件中去吧。 如果你看一眼 build.xml 文件你就会发现， build.xml 文件使用 ant 任务对 eclipseme-build.xml 进行了引用。
注意： 默认的 build.xml 文件假设你使用 混淆器对你的 MIDlet 进行打包。 如果你不使用混淆器的话，你必须对 build.xml 文件做适当的修改。 首先，在 build.xml 文件中找到这行：
<target name="deployedSuite" depends="deployable" />
然后改成这样:
<target name="deployedSuite" depends="jar" />

- 关于对 MIDlet 套件进行数字签名

从 Eclipse 0.7.0 版本开始，开始支持使用密码对 MIDlet 进行签名。 本文档提供了数字签名处理的背景知识，描述了手动对 MIDlet 套件进行数字签名的大概流程， 接着说明了 EclipseME 是如何处理的。

- 背景知识

在 MIDP 1.0 规范中，所有 MIDlet 都使用一个“沙盒(sandbox)”安全模式来运行。 因此实质上，MIDlet 只能只能访问 MIDP 1.0 规范中限定的 API 集，以及打包在同一 MIDlet 套件中的库。
MIDP 2.0 (JSR-118) 把“受信任的(trusted)”和“不可信的(untrusted)”概念引入了 MIDlet 套件。 一个“不可信的”套件运行在与 MIDP 1.0 一样的限制环境中——所有可用的 API 集都只能是 MIDP 规范的一部分。 而“受信任的”套件则可以被赋予访问设备的更多 API 的权限。另外，作为 MIDP 2.0 规范的一部分， 某些“标准”API 也需要授权才能访问。 比如，一个不可信 MIDlet 套件必须能够访问 javax.microedition.io.HttpConnection API， 但是系统需要用户确认是否允许 MIDlet 套件使用这个 API。
而另一方面，受信任的 MIDlet 套件则不需要用户明确授权就可被赋予该 API 的访问权限， 而且可以访问那些禁止不可信 MIDlet 访问的 API。

- 对 MIDlet 套件进行数字签名的基本步骤如下：

1. 在 JAD 文件和 manifest（清单）中包含你的 MIDlet 套件要求获得的权限列表
2. 获取适当的密钥/证书对来进行数字签名
3. 在 MIDlet 套件的部署过程中进行最后的数字签名步骤

- 在 JAD 文件和 manifest（清单）中包含你的 MIDlet 套件要求获得的权限列表

MIDlet 要在应用描述文件(JAD)和 JAR 清单(manifest)中声明其要求的权限， 可以使用 MIDlet-Permissions 或 MIDlet-Permissions-Opt 这两种标签之一。 可以使用上面的标签指定多种权限，之间用逗号分隔。 在安装过程中，设备需要检查 MIDlet 要求的权限。 如果在 MIDlet-Permissions 中声明了某种权限， 那么设备要么就要把 MIDlet 安装在保护域中并赋予其相应权限，要么就必须中止安装。 而如果权限是声明在 MIDlet-Permissions-Opt 标签中的， 且 MIDlet 套件未被授予对适当保护域的访问权限，安装则是可以继续的，但套件无法获得其要求的访问权限。 所以，如果你的 MIDlet 套件依赖于某种权限，不能在没有它的情况下运行， 那么对该权限的要求应该声明在 MIDlet-Permissions 之中。
如果你的应用程序可以使用某种权限，但即使未获得该权限也能运行， 那么使用 MIDlet-Permissions-Opt 来请求访问权限。 如果你的 MIDlet 未被授予对某种特性的访问权限，那么尝试访问该特性会导致抛出 SecurityException 异常（安全性异常）。 值得注意的

是对于 HTTP，某些设备提供商略微的违反了规范的约定，抛出的是 IOException 异常（输入输出异常）， 而不是 SecurityException，这是为了对 MIDP 1.0 保持兼容的临时性措施。 对于这点请与你的无线工具包提供商的文档进行核对。

下面是一个 JAD 文件的例子，其中包含对权限的请求。这个 MIDlet 套件要求必须拥有对 HTTP 和 HTTPS 的访问权限， 并且希望访问 PushRegistry（推送注册），VideoControl.getSnapshot(拍照)， 以及 SMS(短信)。如果后三个被拒绝，套件能够调整且对用户仍有价值。

```
MIDlet-Permissions: javax.microedition.io.Connector.http,
                    javax.microedition.io.Connector.https
MIDlet-Permissions-Opt: javax.microedition.io.PushRegistry,

javax.microedition.media.control.VideoControl.getSnapshot,
                        javax.wireless.messaging.sms.receive,
                        javax.wireless.messaging.sms.send
```

在这个例子里，HTTP 和 HTTPS 不是可选权限。因为应用程序已经申明不能在缺少它们的情况下运行， 所以如果设备不能在安装时授予这些权限，安装就会中止。是否可以进行授权决定于设备的安全策略， 以及 MIDlet 是否是受信任的。为了成为受信任的 MIDlet，就必须对 MIDlet 进行数字签名。

- 获取适当的密钥/证书对来进行数字签名

MIDlet 安全措施基于公钥加密法。基本步骤：

1. 创建一对公钥/私钥。
2. 使用私钥对 MIDlet 套件进行数字“签名”。
3. 接着把公钥作为 MIDlet 套件的一部分来发布，使任何人都可以验证此数字签名。

因为使用你的私钥加密的信息只有用对应公钥才能解密，所以只要你对私钥进行保密， 这样全世界都可以确信，只要是能使用你的公钥进行解密验证的信息，都肯定来自于你。

然而，这个方案仍然回避了一个问题：别人如何知道一个公钥真的来源于你？ 如何防止我自称比尔·盖茨，并以他的名义签署文档？ 因此，不能仅仅把你的公钥包含在 MIDlet 套件中，而是要在 MIDlet 中加入一个包含你的公钥的数字证书。 数字证书是由一个可信任的第三方（如 Verisign）颁发的，来承诺保证此公钥确实是你的。

那么设备为什么能够信任来自 Verisign 的数字证书呢？ 因为设备在开发的时候已经预置了一个“根证书颁发机构(root certificate authorities)”的集合， 他们总是可信的。证书本身是使用手机中预置的信息进行自签名的。 这样，设备就可以使用预置的信息来验证数字证书确实是由 Verisign 颁发的。 另外，你的证书也可以是使用一个“中介(intermediate)”的密钥签署的， 而“中介”的公钥，是使用一个受信任的密钥签署的。这个机制叫做“证书链”。 证书链中可以有多个中介的密钥，理论上链的长度可以是任意的，不过极少有包含多于 2 或 3 个连接的证书链。

因此，设置你的密钥的典型流程如下：

1. 使用工具软件产生一对公钥/私钥。
2. 产生一个包含你的公钥的“证书申请”，并发送到证书颁发机构。
3. 证书颁发机构进行一些调查，以核实你对自己身份的声明与事实相符。 具体进行哪些调查，每个颁证机构的做法不同。
4. 当颁证机构核实你的声明和真实身份确实相符之后（并且向他们交费之后）， 他们会把你的公钥用一个证书包装起来，并发还给你。
5. 你把证书导入自己的密钥库，这样你要对什么东西进行签名的时候就可以随时使用了。

作为 JDK 的一部分，Sun 同时提供了一个工具来帮你管理密钥。 这个工具被恰如其分的命名为 keytool。 使用下面的命令来产生一对公钥/私钥：

```
keytool -genkey -keyalg RSA -keystore <keystore file name>
```

这条命令指示工具产生一对新的 RSA 公钥/私钥对，并把它们保存在给定的“密钥库(keystore)”文件中。 作为此过程的一部分，工具要求你提供一系列关于你本人的信息。 另外，还会询问你下列项目：

Key Alias (密钥的别名)	一个密钥库可以保存多个密钥/证书对。“别名”用来在密钥库中标识特定密钥/证书对。
Keystore password (密钥库密码)	显然你的密钥库中并非人人可以访问。因此，密钥库文件是加密保存的。 这个密码用来获取对密钥库的访问权限。
Key password (密钥的密码)	使用密钥库密码，你可以访问密钥库中“公开”的那一部分——公钥和证书。 而要访问私钥，则必须使用另外的密钥密码。此密码用来对私钥进行进一步的保护。

如果你愿意，可以在 keytool 的命令行上直接指定上面的所有参数。 请运行 keytool -help 或者到 Sun 的网站上参考进一步信息。

如果你只是想“玩玩”数字签名，你可以跳过这一节直接阅读关于签署(signing)的讨论。 当生成你的密钥之后，keytool 自动向密钥库中加入一个“自签署证书”。 这个证书格式正确，可用来对 MIDlet 套件进行签名。 但是记住这并不能使你的 MIDlet 套件成为受信任的成品，因为你的手机无法使用任何一个它信任的颁证机构来验证这个“个人的”证书。 而要想达到那个效果，你仍然必须按照上面的流程来申请证书。 不过，如果你只是想体验一下 EclipseME 的数字签名支持，这个“自签署”的证书倒是足够用了—— 日后你可以等待获取一个“真正的”证书。

要使用 keytool 来产生一个证书申请，可以用下面的命令行：

```
keytool -certreq -keystore <keystore file name> [-storepass <storepass>]
            [-alias <alias>] [-file <request file>] [-keypass <keypass>]
```

在括号中的项目是可选的——keytool 会自动询问你那些未在命令行中指定的项目。

产生你的证书申请之后，就可以把它寄给颁证机构。 当他们把证书发回给你之后，你需要使用下面的命令行把它倒入你的密钥库中：

```
keytool -import -keystore <keystore file name> [-storepass <storepass>]
            [-alias <alias>] [-file <certificate file>] [-keypass <keypass>]
```

此命令行会使用颁证机构颁发给你的、签过名的证书替换掉原来使用 keytool 生成的“自签署”证书。 到这里你就真正准备好使用密钥库中的信息对 MIDlet 套件进行签名了。

- 在 MIDlet 套件的部署过程中进行最后的数字签名步骤

如果你不用 EclipseME，可以使用 JADTool 来签署你的套件。 JADTool 是由 Sun 及许多其他厂商和他们的无线工具包一起提供的。

使用 JADTool 需要两步：首先使用下面的命令把 JAR 的签名加入到 JAD 文件中

```
java -jar JADTool.jar -addjarsig -keypass <password> -alias <key alias>
        -storepass <password> -keystore <keystore> -inputjad <filename>
         [-jarfile <filename>] [-outputjad <filename>]
```

这个命令计算 JAR 文件的校验和，然后使用你的私钥对此校验和进行签名，然后把结果编码到 JAD 文件中。 第二步是把你的公钥证书添加入 JAD 文件，使用以下命令

```
java -jar JADTool.jar -addcert -keypass <password> -alias <key alias>
        -storepass <password> -keystore <keystore> -inputjad <filename>
         [-jarfile <filename>] [-outputjad <filename>]
```

这个命令从密钥库中提取证书，对它进行编码，然后添加入 JAD 文件。 到这里，你的 MIDlet 套件就被完全的签名了。如果你检查签名之后的 JAD 文件， 你会看到其中包括类似下面的两行：

MIDlet-Jar-RSA-SHA1:
VPHnTUqz5+R6G29HTtEfIC4DOpXzpCa9U3zBqe0kGhOitgh1wwdcK4jcQnfjSTD9kPRfheWiIwC8xeCs
O8wdlE9xY/v8veYic0cj6GtSmO3EgL5Mc+KRSNfitVIL7xa5LWY7yTCi7IkiIDqC+dP8KQjoCReGsUOY
oPM9iq6b6dM=MIDlet-Certificate-1-1:
MIICUjCCAbsCBEGjQ2AwDQYJKoZIhvcNAQEEBQAwcDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAkZMMRMw
EQYDVQQHEwpGb3J0IE15ZXJzMRIwEAYDVQQKEwlFY2xpcHNlTUUxFDASBgNVBAsTC0RldmVsb3BtZW50
MRUwEwYDVQQDEwxLZXZpbiBIdW50ZXIwHhcNMDQxMTIzMTQwNDE2WhcNMDUwMjIxMTQwNDE2WjBwMQsw
CQYDVQQGEwJVUzELMAkGA1UECBMCRkwxEzARBgNVBAcTCkZvcnQgTXllcnMxEjAQBgNVBAoTCUVjbGlw
c2VNRTEUMBIGA1UECxMLRGV2ZWxvcG1lbnQxFTATBgNVBAMTDEtldmluIEh1bnRlcjCBnzANBgkqhkiG
9w0BAQEFAAOBjQAwgYkCgYEA1uL1PGDcD/u1Ki12gZntby2lJmizbzDpbLSV30fs5BRO21dC2XY99X7i
goc0+02Ic5qSzY7x2r9rQf0c+d8hA9T3w4tjQBNizEdkvxFNcFFNrAo5MdzWvcUPiCODqthy1O9LrCAo
mp/+2d5N+TGOgtUXocgDPouaT/rfKQujI5sCAwEAATANBgkqhkiG9w0BAQQFAAOBgQBIQ4ECwjj3spYO
hxsih55F7e0Prx+evi6VYBaWoZTGKjevnW3IkKTijytdoMpBX3r7oBjAIibFlSEqUJfJRMszq/L9JOnI
cEKRIvkW8yN/Ls8pWB5VEir2EUh3kiIgk2zo7uhbjs58b5b0jWhTZMhQjPI22I23Tqa/HKXNBsTL3A==

第一行是加密后的 JAR 文件校验和，第二行则是包含你的公钥的证书。 如果颁证机构把一个中介证书随同你的证书一同颁发给你（一个多重证书链）， 那么还会有一个 MIDlet-Certificate-1-2 项目。

对 MIDlet 套件签名之后，你的移动设备就可以使用下面步骤验证你的 MIDlet 套件了：

1. 重新计算产生 JAR 文件的校验和。
2. 验证 JAD 文件中的证书可以一直回溯到设备信任的某个颁证机构。
3. 从证书中提取你的公钥。
4. 从 JAD 文件中提取出当初加入的、加密过的 JAR 校验和
5. 使用你的公钥把校验和解密。
6. 把解密后的校验和与第一步中计算出的校验和进行比较。

如果最后一步的两项匹配，则移动设备可以确信

1. 此 MIDlet 套件确实是由你签署的。如果是其他人签署的，那么解密过程无法成功， 因为你的公钥不会和别人的私钥相匹配。
2. 自从你签署过之后，此 MIDlet 套件未被修改。否则，校验和就会不一样。

最后，很多设备会接着检查 JAR 文件中的 MANIFEST.MF(清单文件) 要求获得的权限是否和 JAD 文件要求获得的权限相匹配。

- 使用 EclipseME 进行数字签名

正如你从上面看到的，把 JADTool 和 EclipseME 一起使用，你需要进行两步处理以部署你的 MIDlet 套件。 首先使用 EclipseME 来构建 MIDlet 套件，然后得使用 JADTool 手动把签名加入到 JAD 文件中。 你每次对 MIDlet 套件进行修改之后都不得不重复这个流程。 每次你改了一行代码并想启动模拟器的时候都得这么做，显然这有些单调。

从 EclipseME 0.7.0 版本开始，它可以帮你完成以往只能用 JADTool 处理的工作了。 使用此功能，请按照下面的步骤：

1. 在 JAD 编辑器的可选属性面板中向 JAD 文件加入你想要的 MIDlet-Permissions 和 MIDlet-Permissions-Opt 项目。 作为部署过程的一部分，EclipseME 会自动把这些项复制到 JAR 文件中的 MANIFEST.MF 文件。

2. 在项目属性对话框中，允许对项目进行数字签名， 并指定要用的密钥库和密钥的别名。完成之后，EclipseME 会自动处理原来需要 JADTool 完成的步骤。 因此，在每次部署的过程中，你的 MIDlet 套件就会被自动签署。

当配置数字签名的时候，最好使用检查设置(Verify Settings)按钮。 这样可以确保密钥库文件的路径，密钥的别名和它们的密码都是正确的。 另外，因为密钥库可以管理多种类型的密钥， 这个操作还会验证用指定的别名标识的密钥是否是可用来对 MIDlet 套件进行签名的正确类型(RSA)。

注意 EclipseME 目前还无法帮你获取用以对 MIDlet 套件进行签名的密钥/证书对。 眼下，你仍然需要按照前面说过的步骤来产生一个密钥，并设法为它申请证书。 不过，只要你把这些信息放入了你的密钥库，EclipseME 就可以从中获取它们。

- 密码管理

我们需要指出的一点是,EclipseME 需要密钥库密码和密钥密码来对 MIDlet 套件进行签署。我们提供了三种方案，你可以用来管理这些密码：

1. 可以设置成让 EclipseME 把这些密码保存为 MIDlet 套件项目的一部分。如果你选择此选项， 你就要在项目属性中输入这些密码。 EclipseME 会把它们保存在项目的.eclipseme 文件中。 为了保护这些密码，它们以加密形式保存。 因此，这可以防止别人在不经意的查看.eclipseme 文件的时候就推导出密码。 但这个方法并非万无一失，因为别人可以访问 EclipseME 的源码以分析密码的加密算法。 不过，在很多环境下，这个方案已经足够安全了，对于“设置并忘记”的做法也很方便， 即使项目是被多个开发者共享的。
2. 一个稍微更安全些的选择是让 EclipseME 把密码保存在工作空间的密码环(keyring)文件中。 默认情况下，这个文件位于
   [workspace>]\.metadata\.plugins\org.eclipse.pde.core\eASEE\org.eclipse.core.runtime\.keyring,
   不过你可以用-keyring 命令行参数来让 Eclipse 使用指定的文件。 如果你使用 CVS 来共享你的项目，此文件是不会被共享的。 Eclipse 自动对 keyring 文件的内容加密(这是 Eclipse 保存你的 CVS 密码的地方)。 你可以使用-keyring 和-password 命令行参数来指定 Eclipse 在非默认位置以非默认密码保存密码环以提高此方案的安全性。
3. 最安全的选项是根本不让 EclipseME 和 Eclipse 保存你的密码，而是在需要的时候提示你输入。 使用这个设定，EclipseME 会在需要密码的时候弹出一个对话框。 为了给你少添点麻烦，EclipseME 会在接下来的会话中把密码保存在内存中， 这样你不用一次一次的重新输入。当你关闭 Eclipse，或是改变工作空间的时候， 这些密码就会被“忘记”。使用此选项，密码绝不会被保存在你的硬盘上。

- 模拟运行

不同的模拟器对签过名的 MIDlet 有不同的表现。 很多模拟器允许你为模拟的 MIDlet 指定一个特定的安全域。 对于支持此特性的模拟器，从 EclipseME 0.7.0 开始， 在设定启动配置中增加了一个项目允许你指定安全域。

注意：如果你使用从颁证机构获得的证书来签署 MIDlet(或者你为了测试而自己创建的证书)，可能有必要把证书导入模拟器以保证你的签署过的 MIDlet 可以正常运行， 尤其是在无线下载(OTA)模式下进行模拟的时候。 另外也有可能你必须使用模拟器支持的证书来签署 MIDlet。请查阅模拟器的文档来获取进一步的信息。

- 参考资料

下面是一些关于数字签名处理过程和与数字签名有关的 MIDlet 属性的有用（但愿）参考资料：

JSR-118

这是MIDP 2.0 的规范。里面包含了数字签名处理的正式规范和多种不同的安全实现。

JADTool说明

此文档来自Sun无线工具包的用户手册。

Nokia的MIDlet签署教程(Nokia Tutorial on Signed MIDlets)

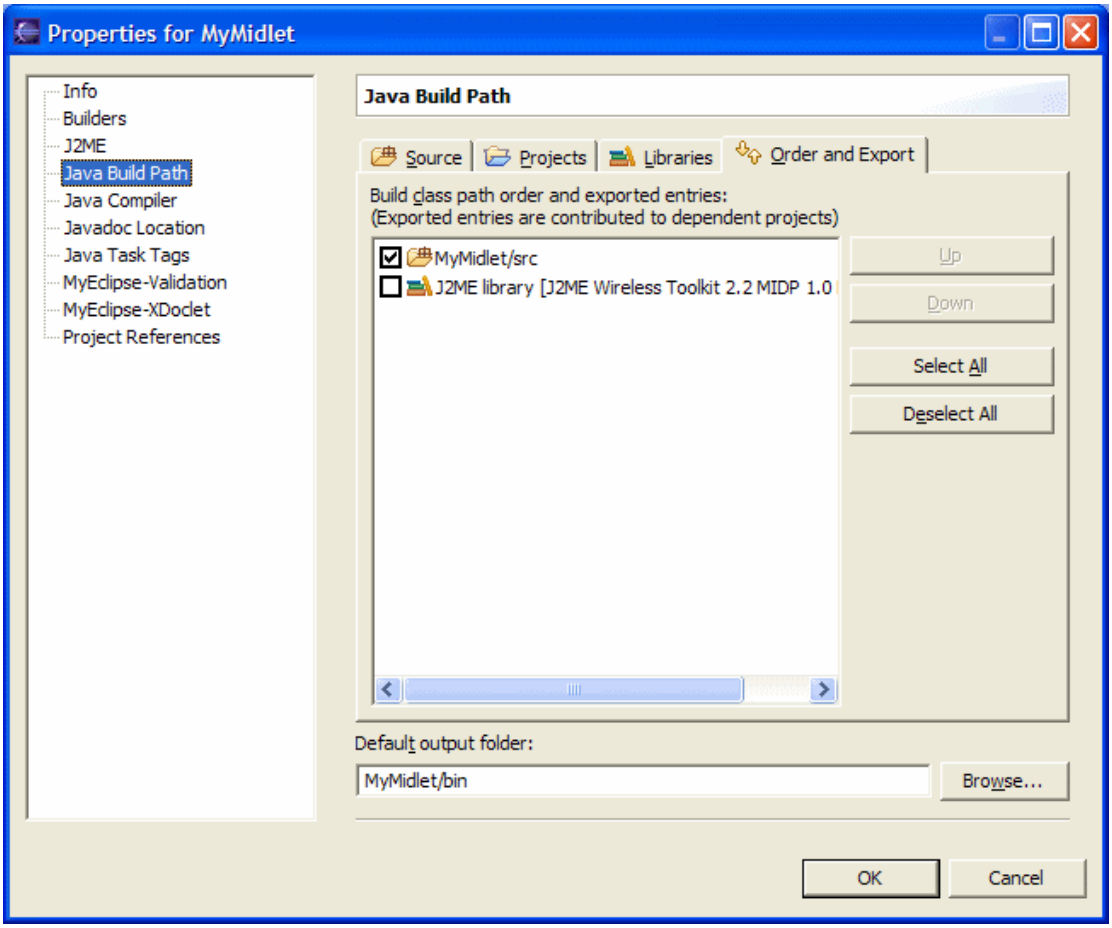
此文档提供了对安全模型和签署流程的简介，并提供了一些例程代码。

13. 从旧版 EclipseME 移植到新版本

该页面将指导你如何从 EclipseME 工程从早期的版本移植到新版本。 除了某些特定的情况，EclipseME 会自动完成这些步骤。

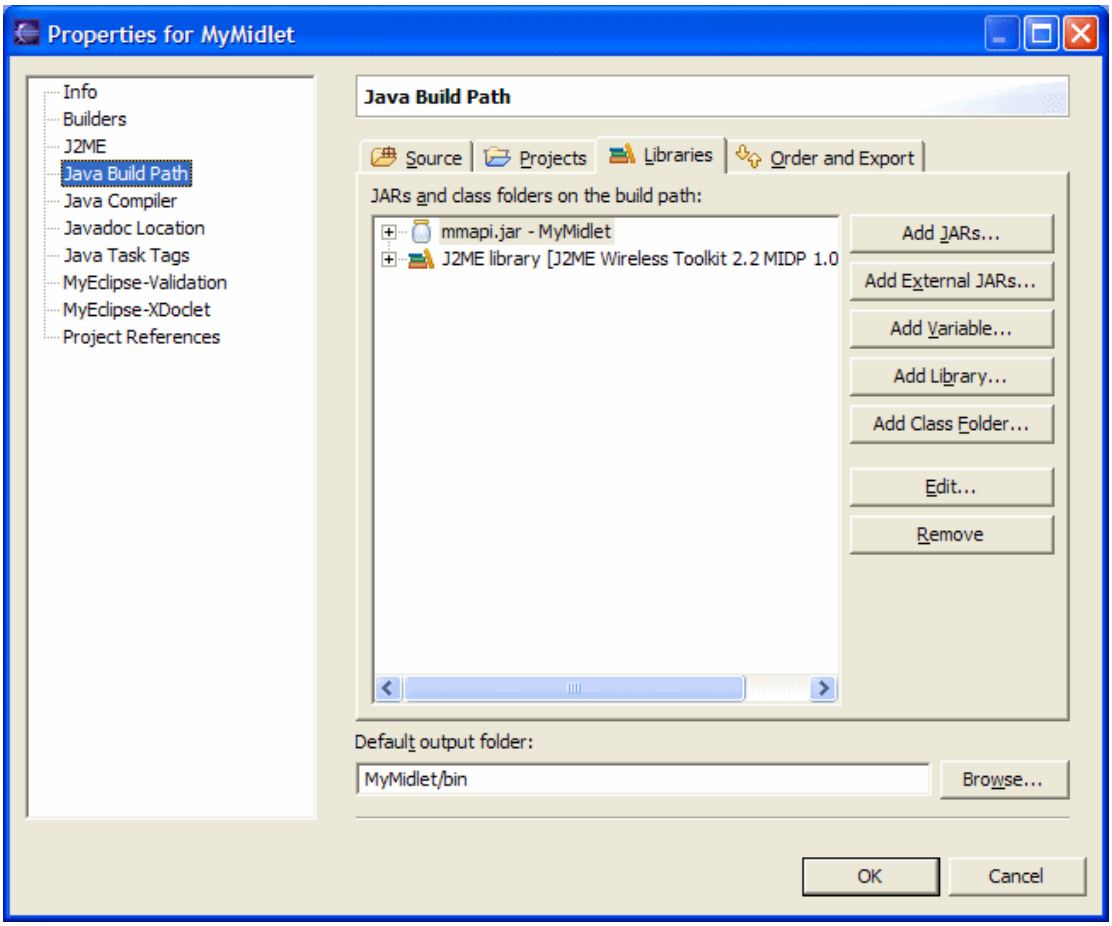
- 向 1.5.0 版本移植
  从早期版本移植到 1.5.0 需要进行手工调整，参见 向 1.5.0 版本移植的指导 一节来获得完整的相关信息。
- 向 1.2.3 版本移植
  向 1.2.3 版本移植是自动完成的。 按照以下步骤可以完成移植：
  1. 打开所有你想进行移植的工程
  2. 退出 Eclipse
  3. 更新 EclipseME .
  4. 启动 Eclipse
  5. 让 Eclipse 完全启动
  6. 退出 Eclipse
- 向 1.0.0 版本移植
  0.9.4 以及更新的版本向 1.0.0 移植是自动的。
- 向 0.9.4 版本移植
  对大多数用户来说，从 0.5.0 以及更新的版本向 0.9.4 移植是自动的。 不过 0.9.4 存在一个改动，使得拥有多个连接在一起的 EclipseME 工程的用户会碰到一些麻烦。

"中断改动"- 归因于 Java 构建路径选项的专家设定 Export Settings of Java Build Path are now honored
背景
在老版本中，如果你的 EclipseME 工程引入了一个外部的包或者需要依赖另外一个 EclipseME 工程， 构建环境会对它们进行递归的引用。JAR 包中的内容或者依赖工程的编译输出都必须经过预校验后 放到当前工程的"verified"文件夹中。当构建的时候，这些经过预校验的资源会被包含到发布出来的 包中。最后，对于 Ant 的用户而言，生成的 Ant 构建脚本中也将会包含这些由 EclipseME 生成的资源。
从技术上讲，这种行为不符合 Eclipse 标准。按照 Eclipse 的标准，只有导出时需要导入其他工程时 才会构成依赖。在使用工程内部的而不是其他项目内的单元测试代码或者其他资源是允许的。
为了从工程中导出资源，可以使用"Java Build Path"选项分支的"Order and Export"选项。

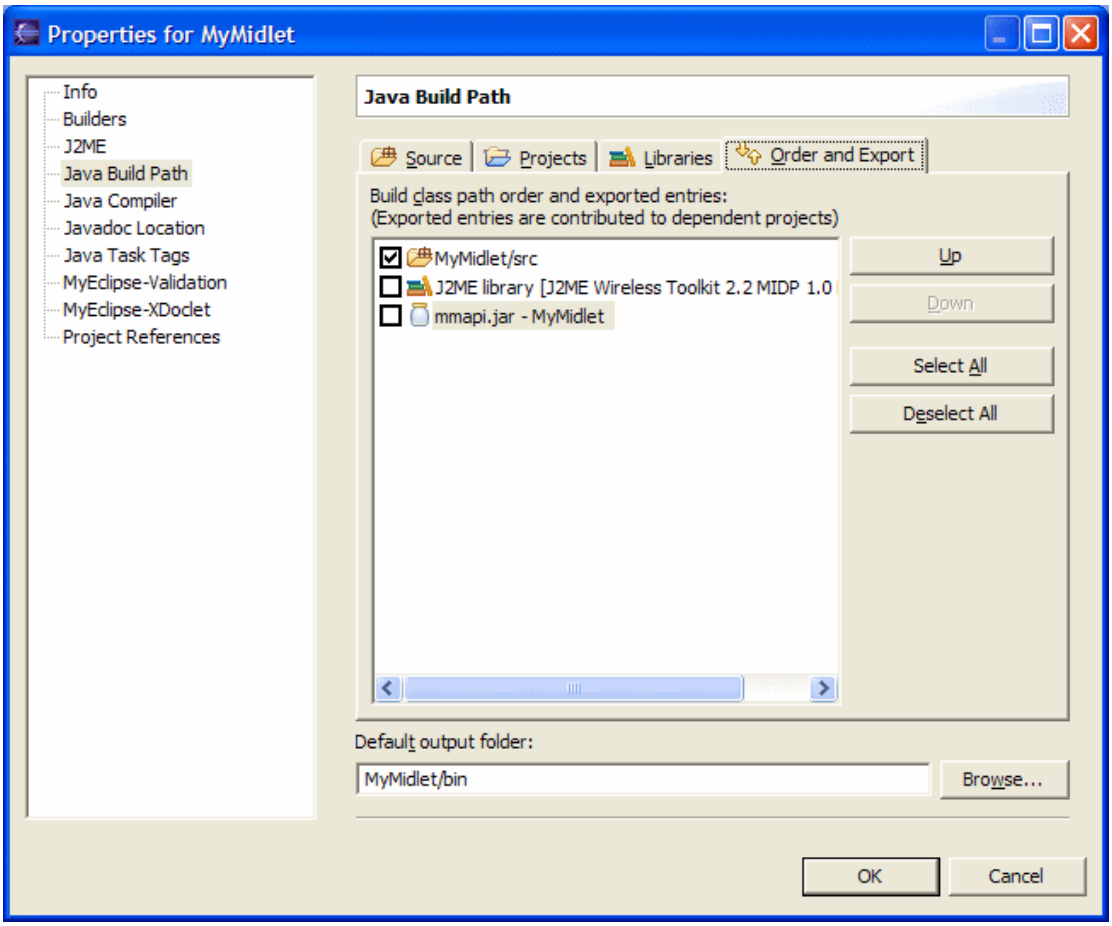


在这个简单的工程中， "MyMidlet/src" 被选中。于是,MyMidlet 的 src 目录中的所有资源被预校验后将 被拷贝到依赖它的项目中。

再来看一个复杂一点的工程。 它包含了一个外部的 JAR 文件。这个文件即可以放在存放工程文件的区域,也可以存放在其他路径下。将这个 JAR 添加进来以后,你就会在工程属性中"Java Build Path"的"Libraries"选项中看到它。



默认情况下，当你为工程添加了一个 jar 包的时候，这个包不会自动的被添加到工程的输出条目中去。 如下图所示：



在 0.9.4 版本之前， "mmapi.jar"中的所有资源会 自动的进行预校核并 被添加到诸如"MyMidlet"之类对它有依赖的包。

在改动以后，将以 java 构建路径选项下的"export"选项框为准。 这意味着，jar 文件和依赖的工程如果没有被显式的选中要求导出，那它不会被预校验以及部署到发布文件中去。 它也不会被添加到依赖该工程的工程中去。 最后，输出的 Ant 构建文件也被设置成以导出选项为准。

对你来说这意味着什么?

如果你的 EclipseME 工程没有包含在发布时需要部署到你的 JAR 文件中去的外部扩展 JAR 包，也没有依赖什么工程，那么这个改动对你没有任何影响，你不需要进行任何改变。

如果你的 EclipseME 工程包含了一个需要发布到部署 JAR 文件的外部 JAR 文件，你必须选中"Exports and Order" 选项页上该外部 JAR 项目的旁的选择框。如果你没有这么做，那么该 JAR 文件中的内容不会被预校验和部署。

如果你的 EclipseME 工程依赖了另外一个工程，那么你必须检查被依赖工程的导出选项是否设置正确。

EclipseME 的开发者对移植中将会产生的一些麻烦表示抱歉。不过为了以后能够让 EclipseME 更好的工作， 对于使用 0.9.4 之前版本并且使用了外部包或者依赖工程的用户造成一些临时的麻烦还是值得的。 特别是当用户引进外部 JAR 文件的时候，可以更加完全更加方便的模拟一个特定设备环境。

向 0.9.2 版本移植

从 0.5.0 以后的版本向 0.9.2 版本移植是自动的。

Migrating to Version 0.9.0

从 0.5.0 以后的版本向 0.9.0 版本移植是自动的。

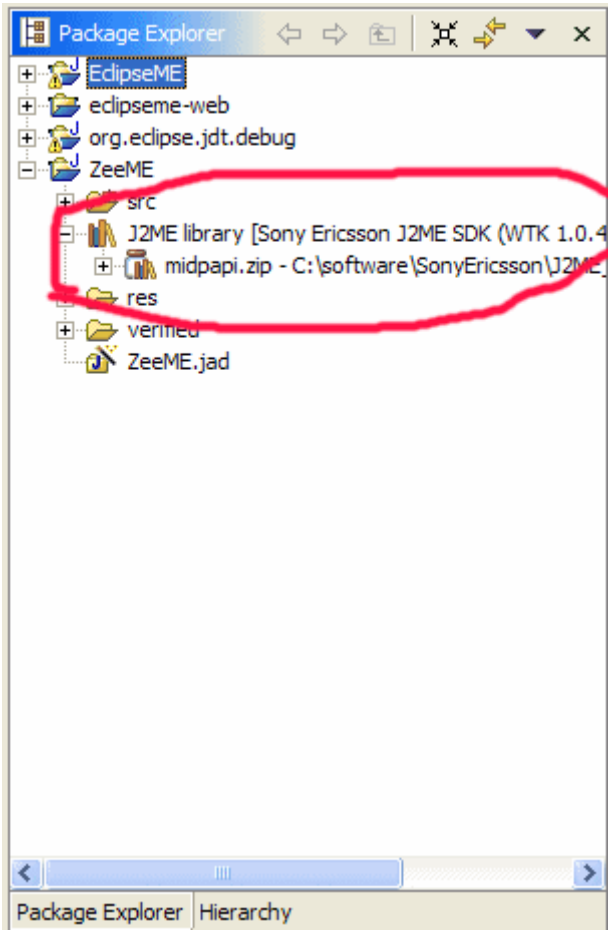
Migrating to Version 0.7.5

从 0.5.0 以后的版本向 0.7.5 移植是自动的。
注意：0.5.0 版本之前的发行版本不支持自动向 0.7.5 移植。所以早期版本的用户向 0.7.5 以后版本移植必须手动调整。
向 0.7.0 版本移植
向 0.7.0 版本移植是自动的。
注意：因为 0.7.0 版本增加了 MIDlet 签署功能，所以 EclipseME 的元数据(metadata) 格式发生了改变。因而被移植到 0.7.0 的版本不能回退到 0.6.1 之前版本。
向 0.6.1 版本移植
向 0.6.1 版本移植是自动的。
向 0.6.0 版本移植
向 0.6.0 版本移植是自动的。
向 0.5.5 版本移植
向 0.5.5 版本移植基本上是自动的, 只有一点需要注意：
按照 RFE 1001717,原来基本 java 首选项 中的"资源"路径选项被去掉了。 如果你原来在这里配置了你的资源目录, 你有两种选择:
去掉资源目录，并且把资源全部拷贝到现有的源代码路径中。
你可以将你的资源目录指定成一个额外的源代码目录。 在 FAQ 中有如何创建一个额外的源代码目录的描述。
向 0.5.0 版本移植
向 0.5.5 版本移植基本上是自动的, 不过为了让移植线程工作你要遵循一下步骤：
关闭 Eclipse
移除原来安装的 EclipseME 插件
将 0.5.0 版本解压到你的 Eclipse 安装目录。该压缩文件必须解压到 Eclipse 的根目录 而不是它的插件目录。
重启 Eclipse
选择并执行一下选项 Project -> Clean... All
最后一步非常重要。大多数的移植工作都在工程的第一次构建时完成。在执行清除(clean) 操作之前要确定所有的 EclipseME 工程都打开了。在移植期间，每个工程的".project"和".classpath" 文件都将被升级以适应插件的新构架。这些文件需要保存到版本控制系统中。
关于安装的更多信息，请参见 安装指导。
向 0.4.5 版本移植
不必移植
向 0.4.0 版本移植
不必移植
向 0.3.5 版本移植
不必移植
向 0.3.0 版本移植
平台组件视图现在 J2ME 首选项中
在 EclipseME 的 0.3.0 版本中,平台组件视图功能被移到 J2ME 首选项中。 在打开包含平台组件的透视图中这会导致一些错误，它期望你检查关于 Eclipse 不能恢复层的错误。选择"ok"会恢复一个不包含平台组件视图的透视图。 这个错误只出现一次。
为了避免错误, 在升级到 EclipseME 0.3.0 之前 .
关闭平台组件视图就可以了。
平台定义存储
在 Eclipse 的版本里,平台定义被保存在一个工程属性中。这个属性不能放到版本控制系统 也不能在其他机器上使用。在该版本中，如果这个属性被发现，它会自动的把它移植到

".eclipseme" 文件中。这个文件可以放到版本控制系统也可以被移到其他机器上而不会出错

从 classpath 变量移植到 classpath 容器

在早些的 EclipseME 版本中，对于每个 MIDlet 套件，平台的的每个库都作为构建路径中的一个单独条目， 这些条目是基于每种无线工具包的环境变量定义，项目一旦创建就很难修改。这一版引入了 classpath 容器的机制来保存源自于选定平台定义的构建路径库。 在下面截图中，MIDlet 套件当前与 Sony Ericsson MIDP 1.0 平台定义相联系。



第一次从 J2ME 项目属性调整 MIDlet 平台定义的时候， EclipseME 会在工程内创建所需的 classpath 容器。 这时项目中任何作为平台定义一部分的单独的 classpath 条目都应该被删除。 很有可能在此次移植后某些单独的条目仍保留着。这些条目应该被手动删除以避免在未来带来问题。

向 0.1.0 版本移植

早期版本的 EclipseME 创建的 JAD 文件格式有错误。 0.1.0 修正了错误格式， 在移植的时候也会自动的更正 MIDlet 套件 JAD 文件的错误。

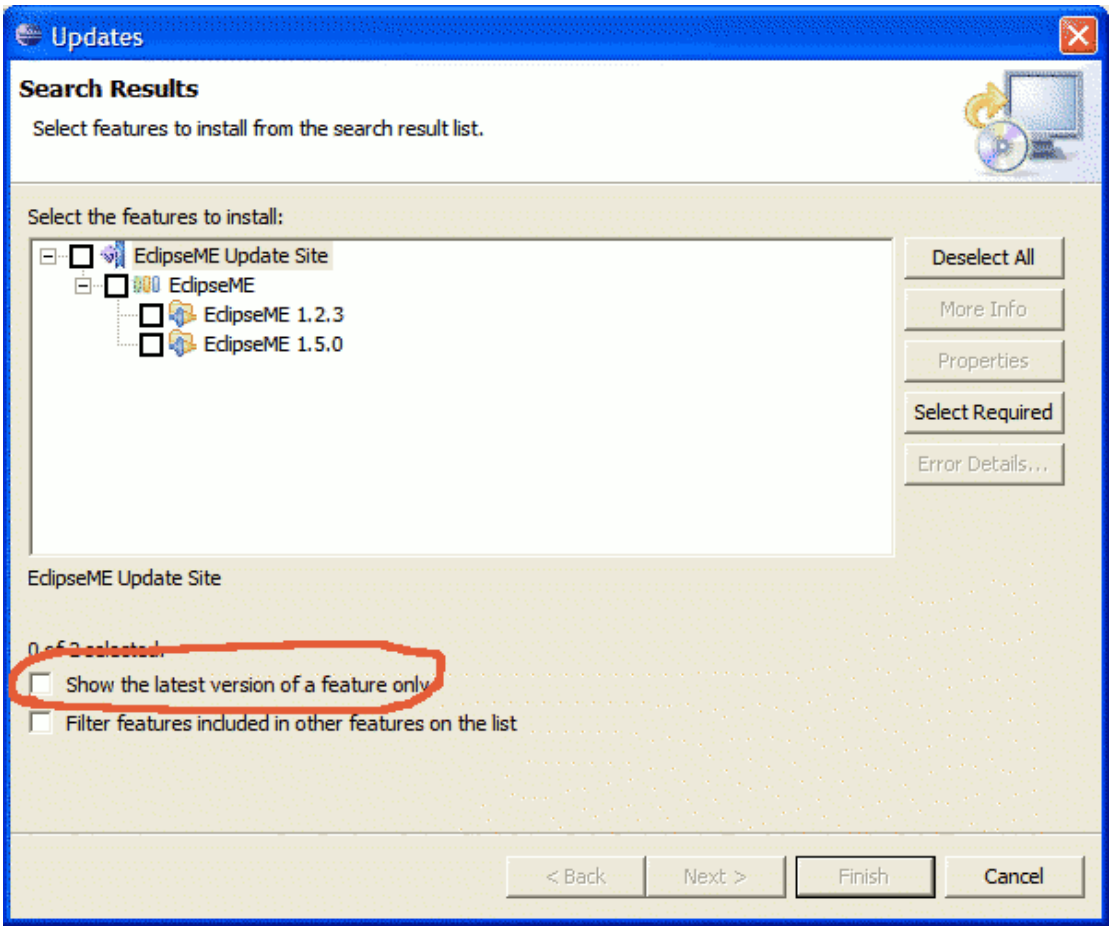
13.1. 移植到 1.5.0 版本

本文档将提供从早期版本移植到 1.5.0 版本的信息。 尽管 EclipseME 为移植提供了有用的帮助，但是仍然需要人的干预。 请仔细阅读本文当来弄清移植过程中所必须的一些步骤。

- 移植到版本 1.2.3

为了正确的移植到 1.5.0,首先你必须至少移植到 1.2.3。首先将你的 EclipseME 升级到 1.2.3,

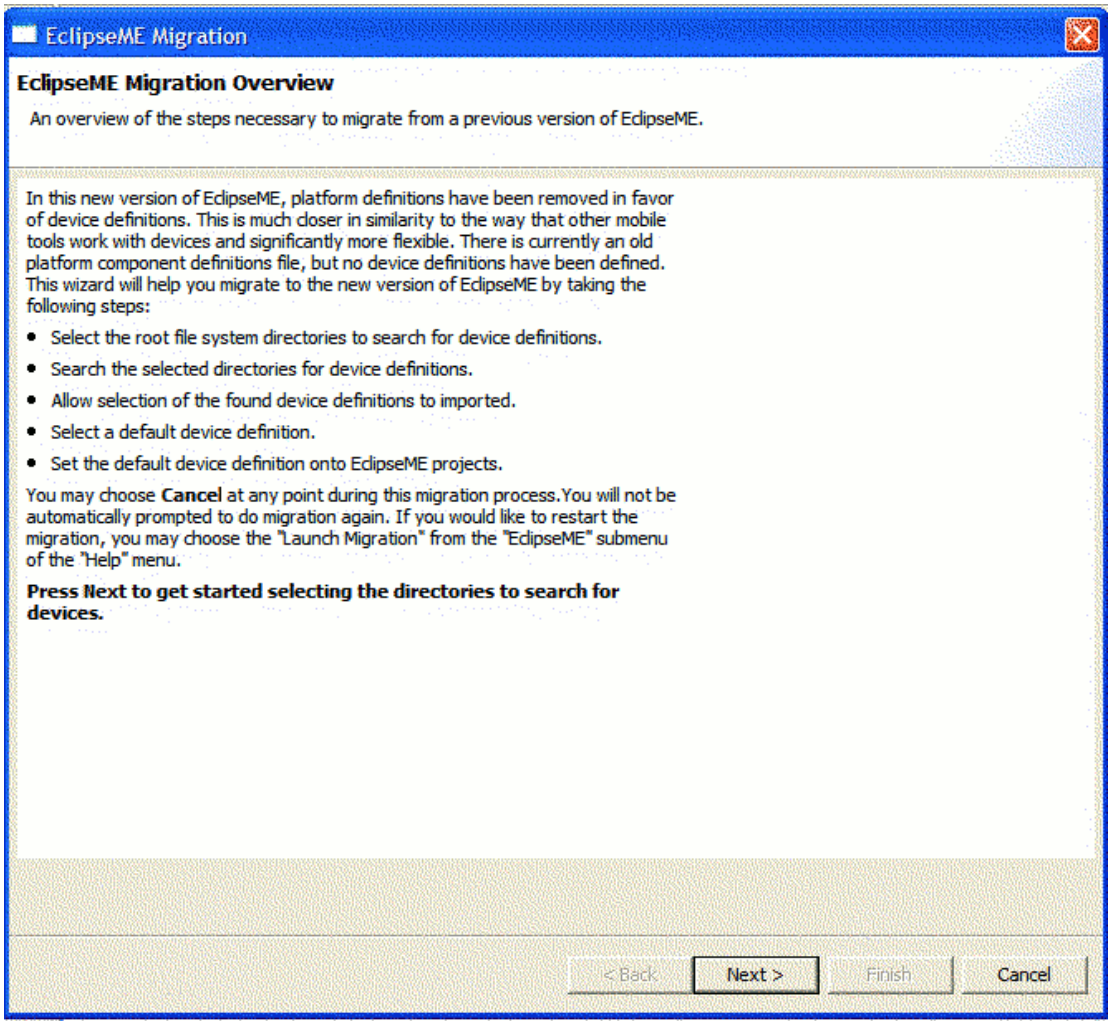
然后按照以下步骤移植：
打开所有你需要移植的 EclipseME 工程
退出 Eclipse 升级 EclipseME 到版本 1.2.3.



当安装可用的 EclispeME 版本的时候，记住去掉 Show the latest version of a feature only 选项的选择。 参见 升级须知一节来获得关于升级的更多信息。
启动 Eclipse
允许 Eclipse 完全启动
退出 Eclipse

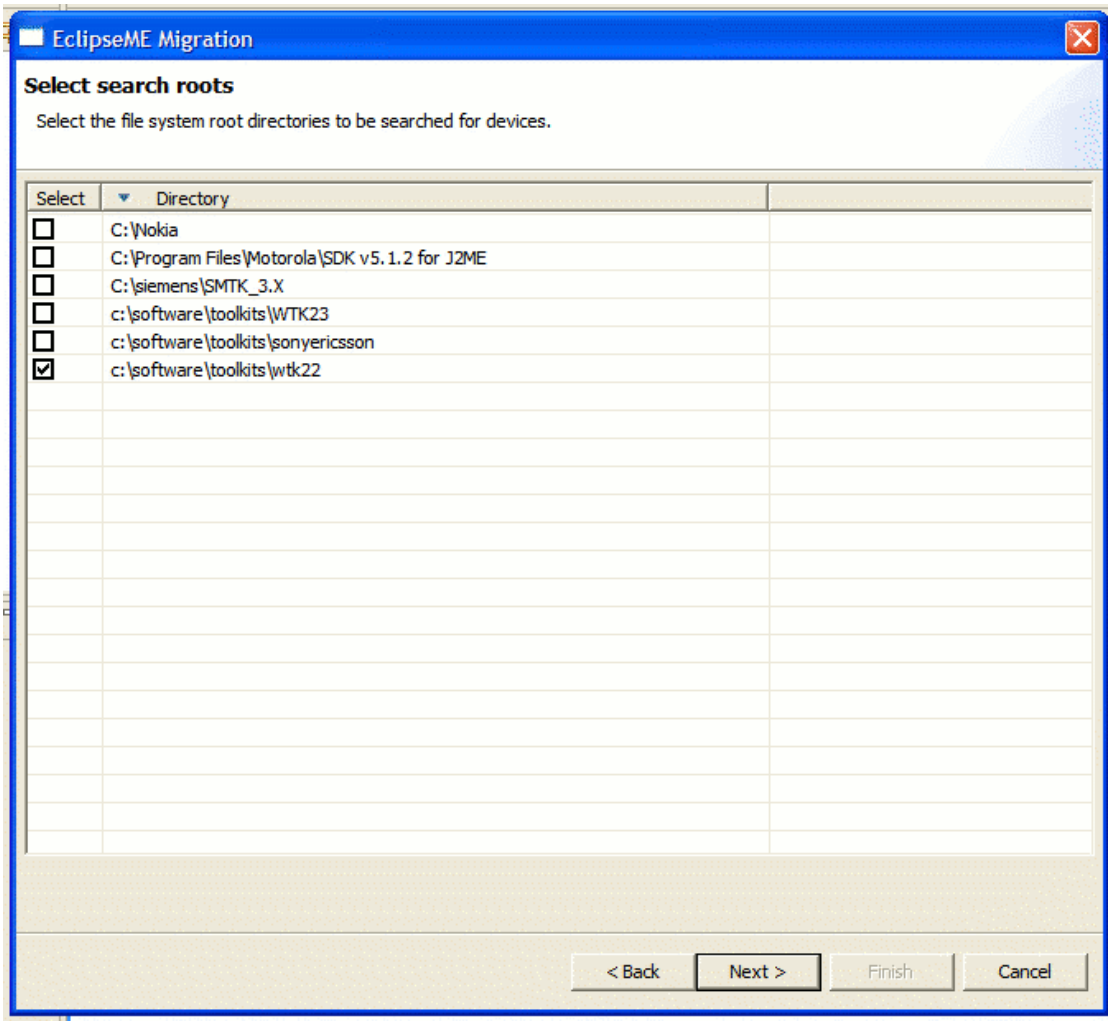
- 移植到版本 1.5.0

从版本 1.2.3 移植到 1.5.0 需要人工干预。Eclipse 第一次打开工程空间(workspace)时 EclipseME 还没有被升级到 1.5.0。 你会看到一个升级向导帮助完成升级的过程。该向导只会在每个还没有被移植的工作空间第一次启动时出现。



选择搜索目录

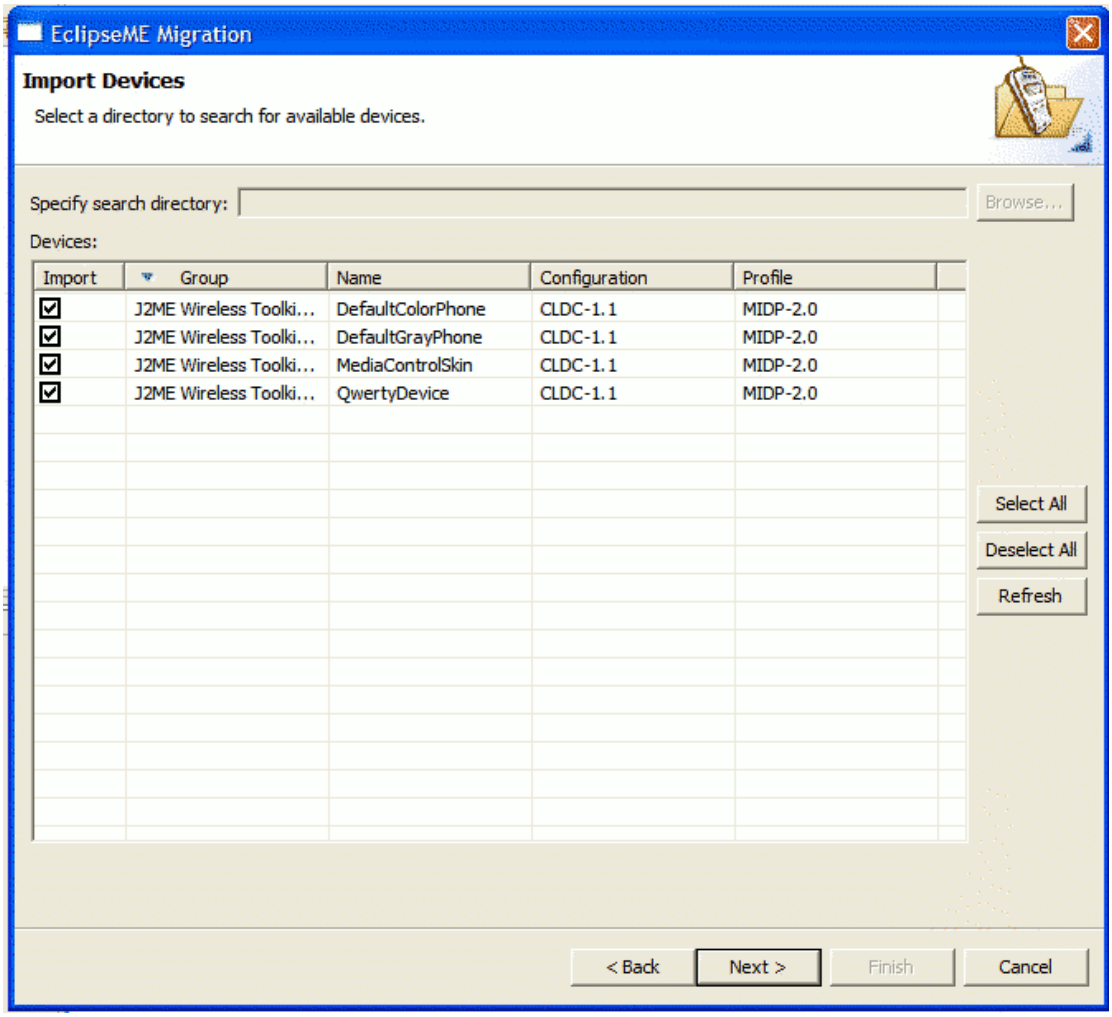
移植的第一步是选择设备定义的搜索目录。EclipseME 会根据前一个版本的配置给出列表。



你希望 EclipseME 在哪些目录搜索设备定义文件，就在目录前面打个勾。

选择导入设备

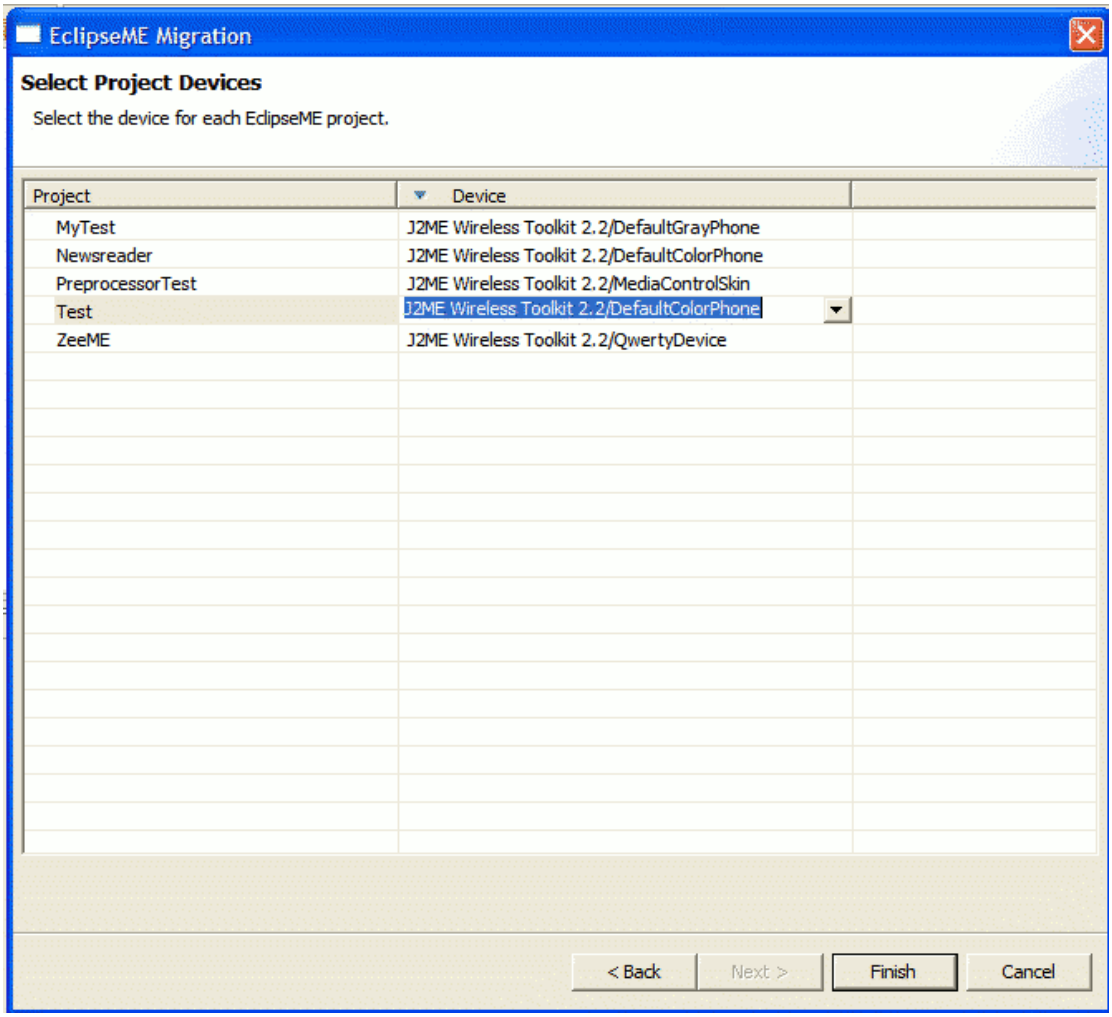
一旦搜索目录被指定，EclipseME 就会在这些目录搜索设备定义。被找到的设备会出现在"Import Devices"页上。



你希望 EclipseME 导入哪些设备定义供运行 MIDlet 使用，就在前面打个勾。

为每一个 J2ME MIDlet Suite 工程选择设备

移植的最后一部是为每个 J2ME 工程选择一个与之关联的设备定义。 EclipseME 会给出一个当前工作空间中 打开的所有 MIDlet 套件的列表，同时还有设备选择框。



使用下拉列表框为每个工程选择你喜欢的设备定义。

完成移植

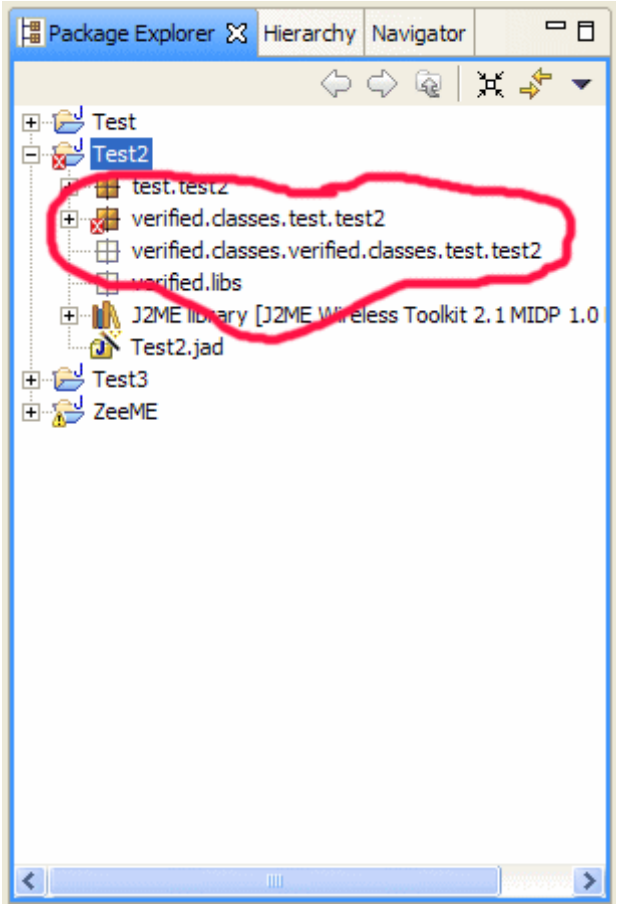
点击 Finish 按钮后移植过程就结束了。 所有选中的设备定义会被导入系统，相应的设备的定义也会按照你的选择与 工程进行关联。

需要注意的是，不管你在哪一步取消了移植向导，EclipseME 不会对你现有的 配置做任何改动。

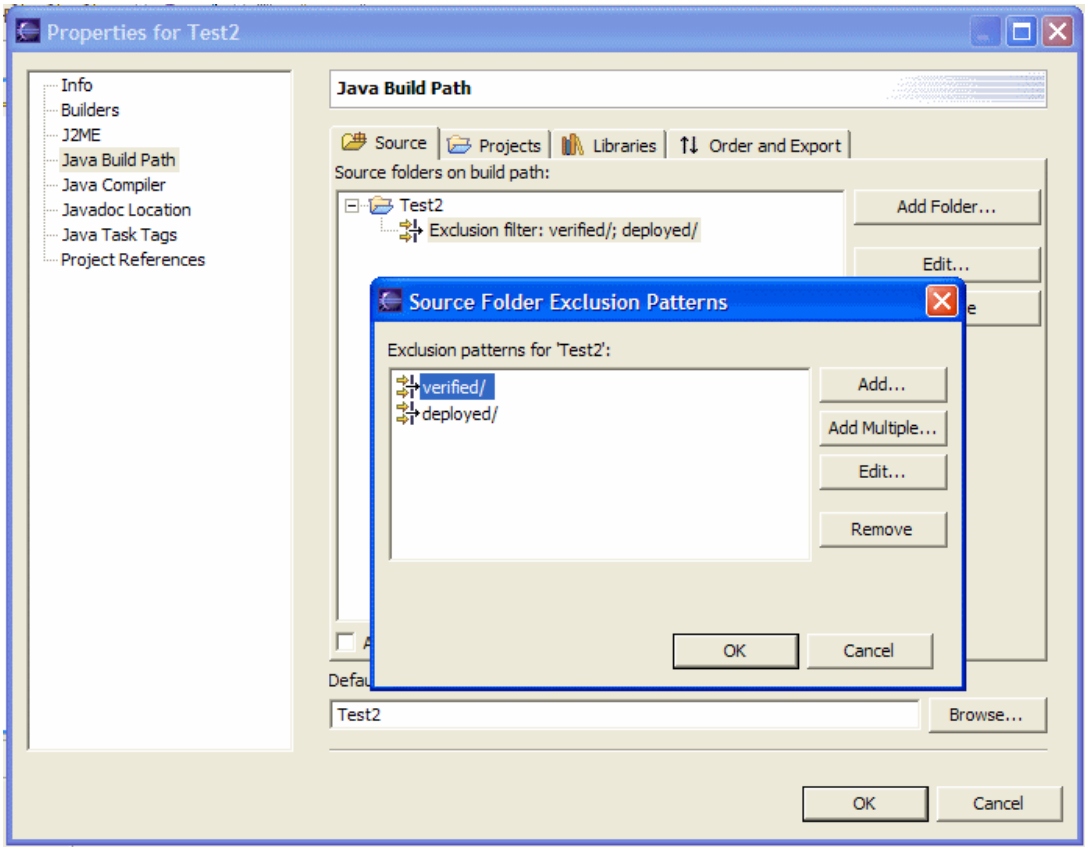
14. 常见问题

- 预校验与部署目录发生错误

如果工程路径与源码目录以及输出目录相同时，预校验与部署文件夹会发生错误。 Eclispe 的 java 工具默认把预校验产生的类也放到 classpath 中，但是它们的路径和 包的结构不匹配，于是就发生了如下图所示的错误。



当这种现象发生时，你必须定义一系列的源码排除样式（Source Exclusion Patterns） 把预校验和部署的文件从 classpath 中移除。你可以照着下图在工程首选项中设置：



在 EclipseME 0.4.0 版本以后，当工程目录与项目目录相同时，这些排除设置会被自动的加进来。

- 不能恢复工作区状态

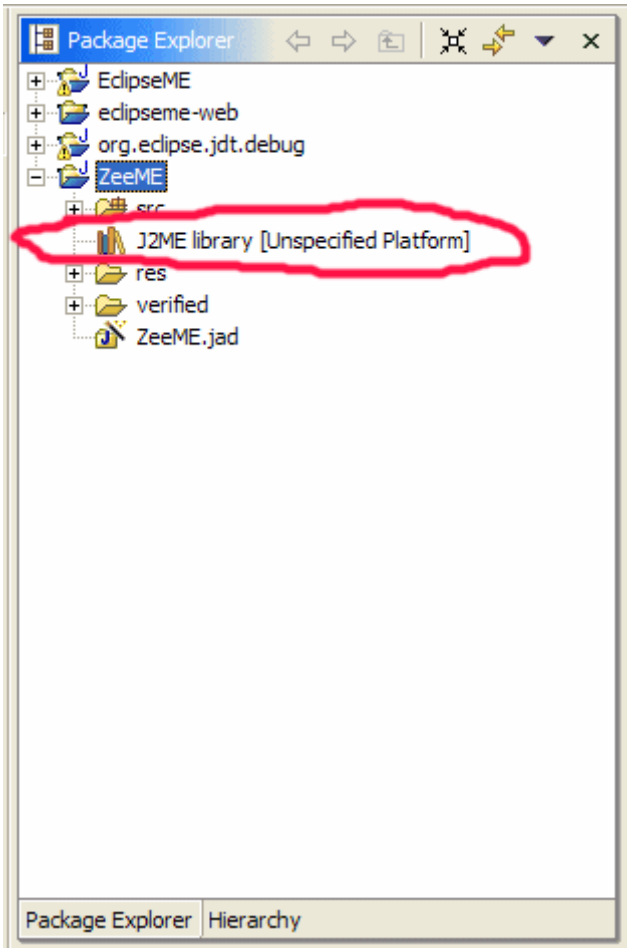
很多因素会造成这个问题，不过最有可能是因为平台组件视图被整合到了 J2ME 首选项中去。这是一个次要的错误，在每个透视图中只会发生一次。参见 migration 来得到详细信息。

- 调试的时候报告"KVM Not Ready"

WTK 的调试器似乎是 Sun 的 MIDP 以及 K Debug Proxy 工具的联合体， 他们被编译到了同一个可执行文件。 通常会有两个不同的工具通过 socket 连接交谈。 当对话中的一半等待另一半的时候，似乎容易给出这个警告。 这个警告会出现在控制台上，但是好像不影响模拟器的调试。

- 为什么我的 MIDLet 套件中会有未定义平台(Unspecified Platform)?

在早期版本中，经常容易把与 MIDlet 套件相关联的平台定义信息搞丢。 这会引起很多问题。如果很不幸它被搞丢了的话，EclipseME 就用未定义平台来代替。



这个工程不包含任何库与信息，你的工程也因而不会被编译。 你可以在 J2ME Properties 中为你的工程关联有效的平台定义。

- 为什么 Eclipse 报告说"build path is incomplete"?

好多原因会造成这个问题，最大的可能是工程的设备平台定义有问题。 参见为什么我的 MIDLet 套件中会有未定义平台(Unspecified Platform)? 的回答。

- 为什么通过命令行签署不能得到类文件

This error occurs because Eclipse doesn't currently request class prepare events for all classes. 参见 EclipseME bug 904266 以及 Eclipse Bug 50531 来获得进一步信息。

- 为什么打包的时候报告"Could not find jar tool executable"

在混淆的过程中，混淆工具产生的 JAR 文件需要被再次预校验。 此时，预处理工具会从环境变量中寻找并调用 jar 工具的可执行文件。EclipseME 一般会在 JRE 安装目录下面去寻找。如果发生这个问题了，肯定是因为 EclipseME 在已经安装的 JRE（java 的子类）中找到了 至少一个 JDK。这种情况下，在 Windows 下，Eclipse 会默认识别 JRE 而不是 JDK。 为了解决这个问题，将已安装 JRE 的路径指定为 JDK 的路径就可以了。 比如在 Windows 上，可以指定类似的一个目录：c:\j2sdk1.4.2 。

- 当我调试我的 MIDlet 的时候，什么也没发生

检查你是不是对调试设定做了必要的配置。 参见 改变 Eclipse 的调试设定来得到详细信息。

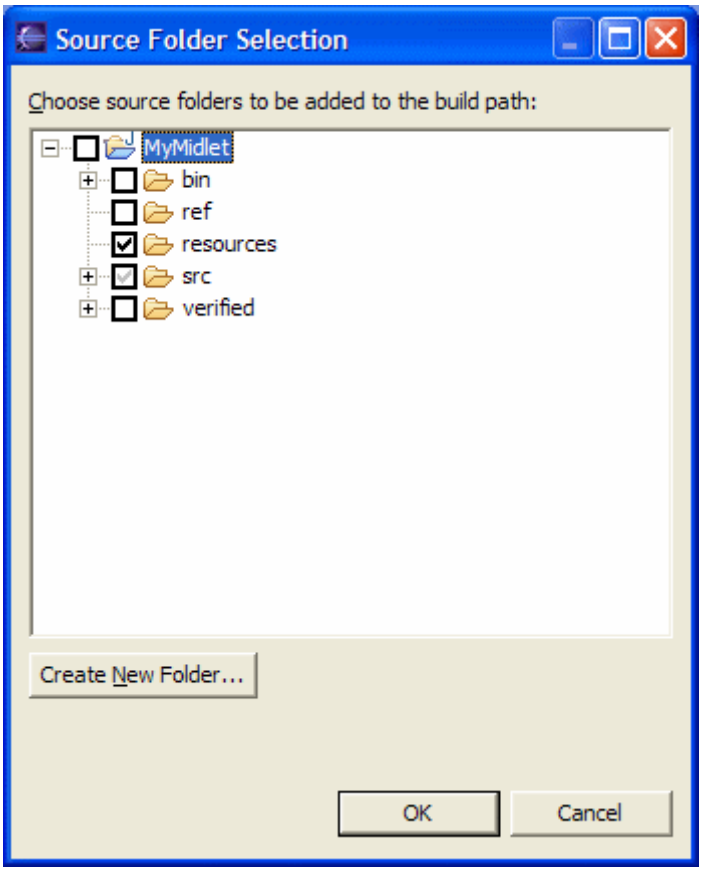
- 当我试图通过 OTA 模式调试我的 MIDlet 的时候，得到如下错误："No midlets defined in JAD File."

你没有使用应用描述文件(JAD)编辑器的 Midlets 面板上定义 MIDlet 可以使用 Java 应用描述文件(JAD)编辑器 为 MIDlet 来添加入口。

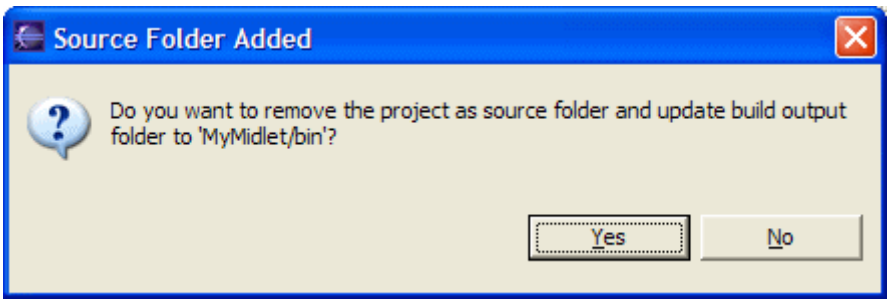
当你手动创建你的 MIDlet 或者 创建新的 MIDlet 中的 添加到应用描述文件选项没有被选中时， 会出现这个问题。

- 我怎么在生成 JAR 文件的时候把图片等资源包含进来?

在 EclipseME 工程中，作为构建的一部分，源文件夹中的的所有 java 文件会 被自动编译，其他的文件会被拷贝到构建文件夹。这是标准的 Eclipse 行为。
可以这样指定一个新的源文件夹:
双击工程或者从工程 / 属性菜单项 选择 Properties 打开工程属性对话框。
从左侧选择"Java Build Path"一栏后再从右边选择"Source"面板。 点"Add Folder..."按钮，然后从工程目录中选择你的资源文件夹， 把它左侧的勾打上表示你选择了它。接下来点 OK 分别退出这个对话框和上层的工程属性对话框。



在你执行这项操作的时候，也许会碰到这个对话框：



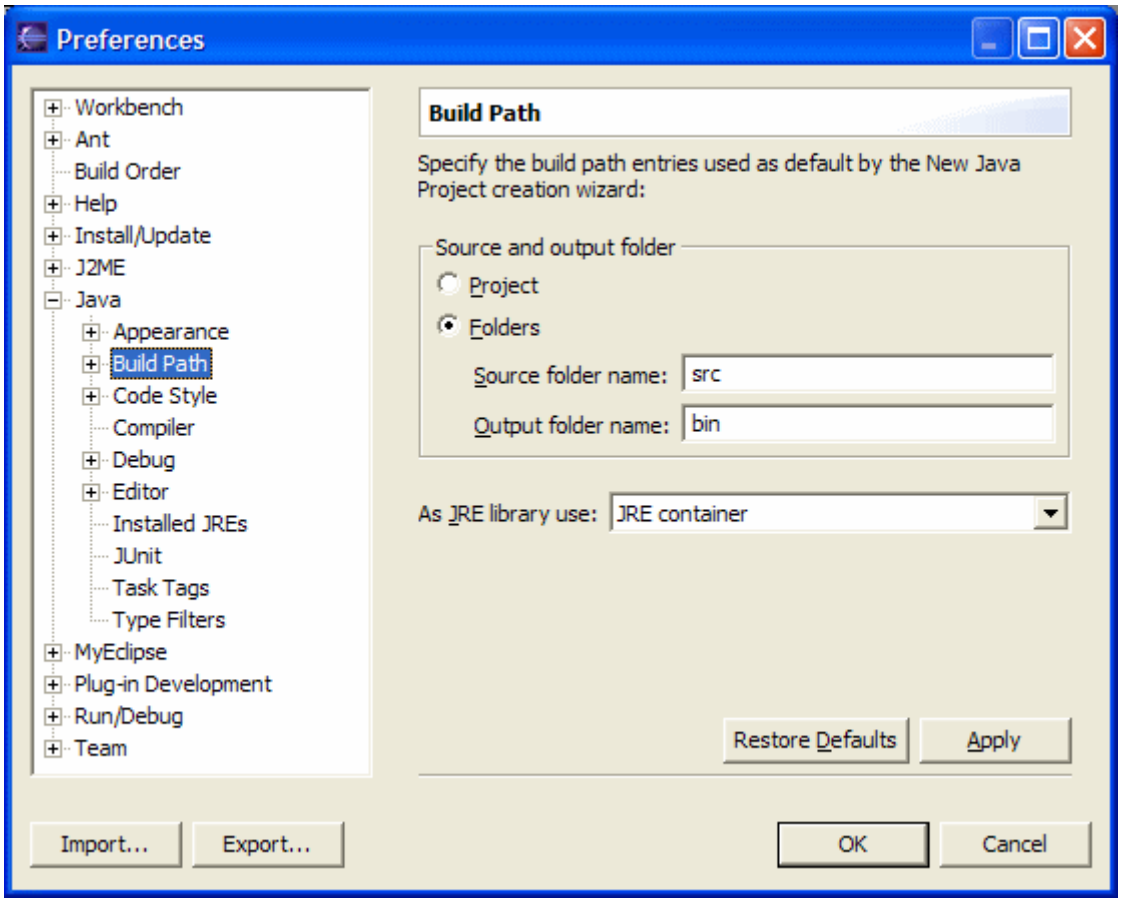
Eclipse 有两种方式来构建新工程：第一，使用工程目录的根目录做为源目录以及输出目录； 第二，使用不同的目录来存放源文件和输出文件，通常这两个文件夹被称作 src 和 bin。你遇到这个问题通常是因为你使用第一种方法创建了这个文件夹。 Eclipse 不支持相互嵌套的源目录，所以当你在一个源目录中(比如工程目录)指定了另外一个 源目录，你就会遇到这个问题。

如果在这种情形下还想将源目录和输出目录分开的话，你就得重新组织你的工程了。 方法时:创建一个叫"src"的新文件夹，将所有的 java 文件转移进去，再按前面说的方法将 这个文件夹(以及你的资源文件夹)指定为源目录。这时，Eclipse 会建议你将 输出目录转换为

"bin"目录，并且询问你是否删除原来的输出文件。对每个问题都回答"Yes" 之后你的工程会被恰当的进行重新组织。

注意：作者认为即使没有单独的资源文件的时候，将输出 与输入文件分离也会比将他们合在一起好。不过 Eclipse 菜鸟们(the Eclipse folks) 肯定不这么认为，因为这不是默认选项。

Eclipse 中有一个设置可以控制在新创建一个工程的时候是否为源文件和输出文件使用不同的 目录。你可以按照一下的步骤来设置：选择 Windows / Preferences 打开 Eclipse Preferences 对话框。 展开左侧的 java 项目，然后选择 Build Path ， 在左侧点 Folders 后会看到这样的画面：



如果你不想使用 src 以及 bin 作为你的源文件目录和输出文件目录的名字，你可以文本框里修改。

记住，这个设置只有在新建工程的时候才有用。如果你想重新组织一个已经存在的工程的话，只能安装前文提到的方法手工调整了。

- 我怎么与其他的 Eclipse 工程共享类文件?

对于标准的 J2SE 应用，一个单独的应用通常是由好多的 JAR 文件组成的。不过在 J2ME 中不是这样。按照 J2ME 平台的要求，所有的类都是 MIDlet 套件中的一部分，发布到真机的时候，它们最终被打到一个 JAR 包中。

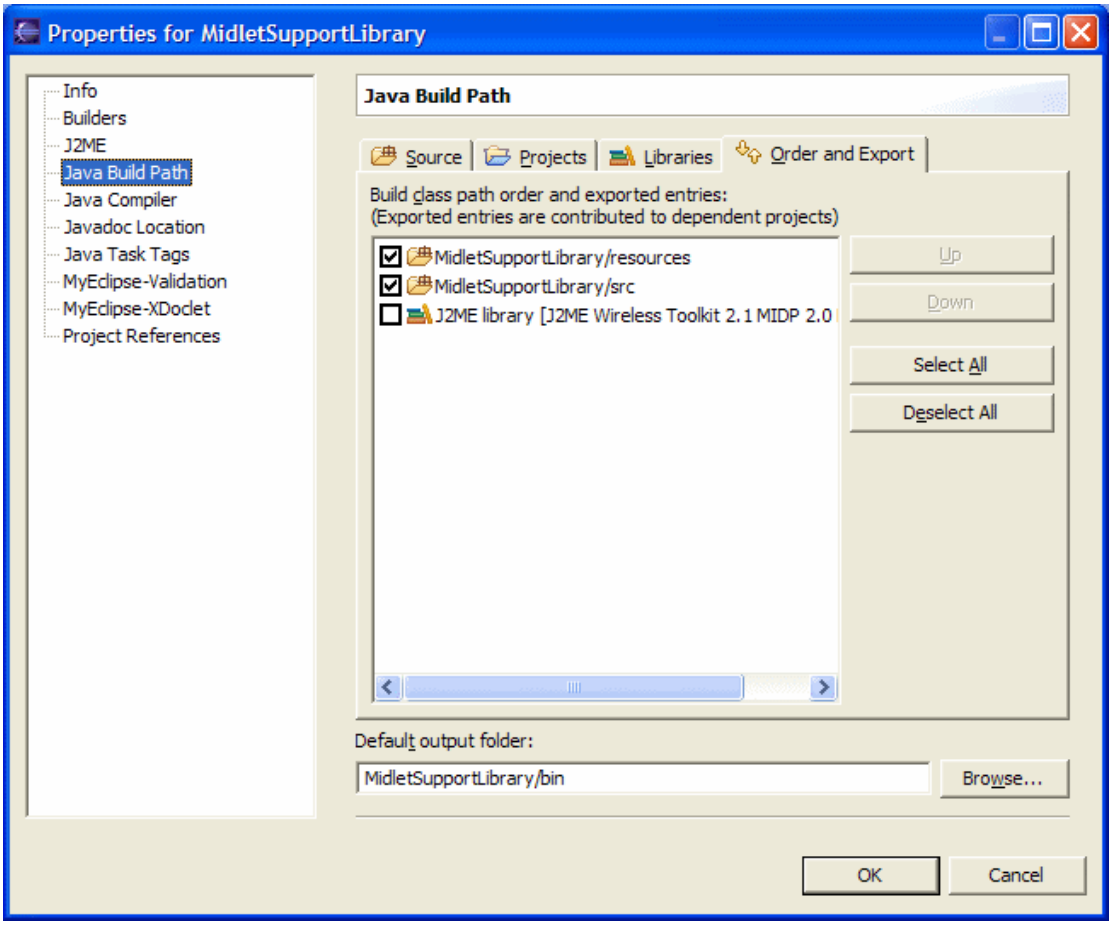
这并不意味着你 不能 mean 在不同的 EclipseME 工程之间共享代码。EclipseME 时可以处理存在依赖关系的工程的。 做为部署过程的一部分，EclipseME 把把所需的类从"子"工程里面抽取出来放到你的 MIDlet 套件中。

为了达到这个目的，你需要按照以下的步骤来作：

所有使用 J2ME 类的工程都需要转换成 EclipseME 工程。不要直接手工将 J2ME 库添加到构建路径， 即使只是工具库——让 EcliseME 来完成这些。 构建路径中所有 J2ME 平台定义中不

存在的库都被假设成在真机上是不存在的。它们在预校验 后将被打包到到你的 MIDlet JAR 文件中去。很明显，你不希望构建进程为了预校验那些真机上 已经存在的库而浪费时间，也不希望他们不打到 JAR 包中。

"子工程"(比如那些类会被"父"MIDlet 包含的工程)必须确保指导出它们自己的类，而不要把 J2ME 的类也一起导出了。看下面这这个例子，source 和 resource 文件夹中的内容被选中导出， 而 J2ME 的类却没有。



EclipseME 会为每个 EclipseME 工程添加合适 J2ME 类，所以你没有必要导出它们。如果 你将它们导出，可能反而会遇到不可遇见的异样，比如当两个工程使用不同的平台工具时。 下面是一下可能产生的问题：

"父工程"绑定了错误版本的类。(因为"子工程"中的类替代了原来的)

EclipseME 花时间为"子工程"的 J2ME 库进行预校验，其实这是不必要的。

EclipseME 将"子工程"的 J2ME 库打包到最后发布的 JAR 文件中。

- 为什么“资源目录”（resource directory）功能没用?

"资源目录"只有在创建工程的时候有用。当工程已经创建完了，在修改这个属性 是没有用的。另外，只有你在 Eclipse 中设置了分别创建源文件目录和输出目录时才有用。 参见安装知道中安装之前一节关于如何配置 Eclipse 的说明。

- 为什么真机说我的应用是非法的?

以下是一些可能的原因：

你的程序可能需要 MIDP-2.0 或者 CLDC-1.1 的支持，而你的真机却只支持 MIDP-1.0 or CLDC-1.0。 确认 JAD 文件是否正确，是不是为你的工程属性选择了合适的平台定义。

你使用了某些设备支持的扩展 JAR，但是另外的真机上却没有这个 JAR。 为你的 MIDlet 套件

添加 JAR 一节有关于这个问题的详细讨论。
你使用了真机不支持的 Java 类文件格式来编译 java 类。 最佳实践 一节有关于这个问题的详细讨论。
当然，以上的列表并不完全，不过已经涵盖了问题的大多数情况。

- 为什么 JAR 或者依赖项目中的类在发布文件中没有被包含进来?

从 EclipseME 0.9.4 版本开始，为了在部署时包含进依赖的 JAR 库，你需要在工程属性的 JAVA 构建路径选项 的定制与导出(Order and Exports)页面中，手工选择要导出的项目。 类似的，在部署时要添加进来的依赖工程中的资源也要如此配置。 为你的 MIDlet 套件添加 JAR 一节有关于这个问题的详细讨论。

- 为什么模拟器在调试的时候崩溃了?

如果你的 MIDlet 套件在运行的时候没问题，但是调试的时候却崩溃了， 可能是你的调试器选项有问题。 确保“发生未捕获的异常时暂挂执行”选项没有被选中，就好像 安装指导中描述的那样。

15. 提交问题报告

- 收集基本配置信息

请收集以下基本信息:
你当前运行的 Eclipse 的版本。选择 Help / About Eclipse Platform， 然后记录下版本以及编译号。
EclipseME 插件的版本号。 从 About Eclipse Platform 对话框单击 Feature Details 记录下 EclipseME 的版本号。
运行 EclipseME 的平台，比如是 Windows XP 还是 Debian Linux。
运行 EclipseME 的窗口管理系统，比如是 Windows XP(C)，GTK 还是 Motif。
你使用的 JDK 的生产者以及版本号。(比如 Sun JDK 1.4.2_03)
这些信息对复现 bug 有极大的帮助。

- 给出如何复现问题的详细描述

你的描述越详细，越具体越好。 一步一步的描述你怎么操作，期望获得什么结果，实际发生的是什么。 特别要注意的是，在描述中最好尽可能多的提供你的首选项设置。 如果代码中有错误，那么首先得可以将错误的代码隔离出来，也就是说 我要可以执行你执行过的操作。这听起来很简单，不过很多时候是你的特定操作序列或者是设置项相互之间 不明显得作用触发了问题。

- 包含工作空间 log 文件

正如你所知道的，Eclipse 将设置以及其他状态信息保存在你的工作空间中。 默认情况下，你的工作空间在 Eclipse 安装目录的 workspace 目录下。 你可以在调用 Eclipse 的时候使用 -data 命令行选项制定另外一个工作空间。
在工作空间目录下，有一个.metadata 子目录。 它可能会包含一个.log 文件。 unix 的用户要记住这个文件默认情况下是隐藏的。
.log 文件包含了有价值得调试信息，包括异常路径。 分离问题的一个办法就是关闭 Eclipse，删除这个文件然后启动 Eclipse，重复你的问题， 接着关闭 Eclipse。 看看.log 文件是否被创建了，如果是将它包含的你的报告中去。
如果你加载了插件开发环境，同样的信息可以通过打开"Error Log"视图获得，这更简单。 如果你没有加载 PDE，这个视图是不可用的。

- 分离出执行的命令行

有时候需要知道 EclipseME 为特定的 WTK 生成了什么命令行。 当你启动 Eclipse 的时候在命令行加上以下参数：
-vmargs -Declipseme.dump.launch=true
它会在.metadata 日志文件中产生额外的日志信息。同样也将它包含到你的报告中来。

- 直接使用 WTK

EclipseME 开发环境又几个部分组成。 Eclipse 是 IDE。EclipseME 是一个插件，它 IDE 和无线开发套件(WTK)提供了桥梁。 WTK 是由不同的提供者提供的 J2ME 的开发环境。 EclipseME 的宗旨是让你可以在 Eclipse 中轻松的开发 MIDlet 套件，而不是直接使用 WTK。
EclipseME 所模拟的不同的设备特征，并不是 EclipseME 本身所实现的。 所有的这些都是由 WTK 来实现。很多情况下，不通过 Eclipse＆EclipseME 来模拟你会得到 更多可以解决问题的信息。只要简单直接通过 WTK 来运行你的 WTK 就可以了。
EclipseME 没有提供将 MIDlet 加载到任何设备上的直接支持，它只是允许你通过 Eclipse 调用 WTK 的编译以及构建的功能。EclipseME 其实并不晓得设备的存在，它只知道它调用 的工

具。

EclipseME 的 OTA 运行特性就是一个好的例子。它只是和 WTK 之间建立了一个连接，从而使用 WTK 的的 OTA 功能，就好像一个真实的设备那样。 当你调试 MIDlet 的时候，直接下载还是 OTA 下载没有太大的差别。不过能够检验 JAD 是否正确， 并且模拟一些只有此时才会出现的特殊问题。

- 收集预处理信息。

新的内置的预校验程序正在开发当中，它还没有被最终的完成。 如果你使用时遇到任何错误，我们欢迎你给我们提供任何信息，这会帮助我们对它进行改进。 在你遇到错误时，对于错误复现的描述越详细，你可以回答的提问越多，那么你提供的信息 帮助也就越大。 在预校验器成熟之前， 我们邀请你到用户邮件列表 来进行讨论。所有的用户都会在讨论中受益。

- 提问

使用该代码产生了什么错误?
错误在什么模拟器或者设备上发生?
错误在多种设备上发生还是特定设备上?
使用外置的设备预校验程序也会出错吗?

- 提供信息

上面问题的答案
会引起问题的源代码
会引起问题的类文件
引起问题的类预校验时的输出

- 检查它是否被报告过了

尽管任何开发者宁可bug被报告多次也不愿没人报告，但是在你报告bug的之前还是请你检查一下 SourceForge 上的当前bug列表，看看是不是已经有人报告了同样的问题。 我已经尽力将公开的bug搞得很短，所以阅读这份列表不会花你太长的时间。如果有人报告了同样的问题，那么它很有可能已经正在被修正了。

16. 技术支持

和大多数开源项目一样，EclipseSE 由开发社区提供技术支持。 无论你是遇到问题，还是想报告 bug，或者想提出建议，我们都会尽自己所能帮助你。

- 提出问题

如果你碰到了问题，最好的办法是将它发送到 用户邮件列表， 这个列表由 SourceForge.net 提供。 EclipseME 开发团队和它的很多用户都在关注着这个列表，并且会尽力解答你的问题。
d
需要说明的是，如果 EclipseME 有了什么异常行为， 在发送问题之前，请按照提交问题报告的步骤 准备有关内容。你收集的信息对我们给你提供帮助很重要。 如果，你跳过该步骤直接，也许你得到的第一个回复就是请你首先按照 提交问题报告的步骤来做。 所以为了节省你的时间，还是请自觉提前完成该步骤。

- 要求改进

如果你有什么改进 EclipseME 的好点子，请提交改进需求到 这里。 我们一直都在寻找下一个 EclipseME 要支持的特性。 你可以看一下 EclipseME 的更新历史， 它多数的改进需求都是来自于用户社区。
在提交改进需求之前，希望你可以使用 用户邮件列表 来讨论一番。 我们使用 SourceForge 的 RFE 列表 作为我们的“待完成工作”仓库。 如果你意外的发现这个列表还没有被全部完成，恐怕它永远都不会有尽头。

- 提交 Bug

我们希望代码是完美的，不过我们只是凡人。
如果你发现了 EclipseME 的 bug，请你一定要发送到 这里。 和上面提到的一样，你要执行提交问题报告的步骤，并把结果包含到 你提交的报告中来。 这会大大减少我们定位问题所在而花费的时间。

我们承诺尽快回复问题，改进要求以及提交的 bug。 不过你要明白，EclipseME 只是我们的爱好，不是我们的日常工作。 我们还有爱人、家庭等。 所以，在你得到回复之前恐怕得耐心等待几天而不是几个小时。
你真诚的朋友,

EclipseME 开发团队

17. Advanced - Developer Documentation

If you're an EclipseME user, this area isn't really intended for you. This area is intended for persons interested in contributing to the ongoing development of the EclipseME plugin.

PLEASE NOTE: As of version 1.5.0 this documentation is no longer up to date. Many of the concepts still loosely apply, but the specifics are no longer correct. Look for a later update to this documentation.

Browsing the EclipseME Code

A description of how to generically access the EclipseME CVS archive can be found here, and you can browse the contents of the EclipseME CVS archive by clicking here.

Downloading the EclipseME Code

If you're going to do any development on EclipseME, the best way to download the code from Sourceforge is to use the built-in CVS support within Eclipse.

Creating the EclipseME CVS Repository Location

Refer to the Eclipse documentation for further instructions on creating a new CVS Repository Location within the CVS Repositories View. Specify the following information when creating the repository location:

Item	Contents
Connection Type	pserver
User	anonymous
Password	(leave this empty)
Host	cvs.sourceforge.net
Repository Path	/cvsroot/eclipseme
Port	Use Default Port

Checking Out the EclipseME Project

The EclipseME project consists of several sub-projects in Eclipse. In general you will want to check out the entire set in order to be able to rebuild the full package.

Expand the newly created EclipseME repository location.

Expand the HEAD branch within the EclipseME repository location.

Right-click on the project or projects that you wish to check out and choose Check Out or Check Out As... from the context menu.

Eclipse will then check out and build the selected projects.

The "Anatomy" page provides a brief description of each of the projects.

Debugging the EclipseME Code

The code may be edited and debugged using the standard Eclipse mechanisms for working with Eclipse plugins. Refer to the Eclipse Plugin Development Environment (PDE) help for further information on developing plugins for Eclipse.

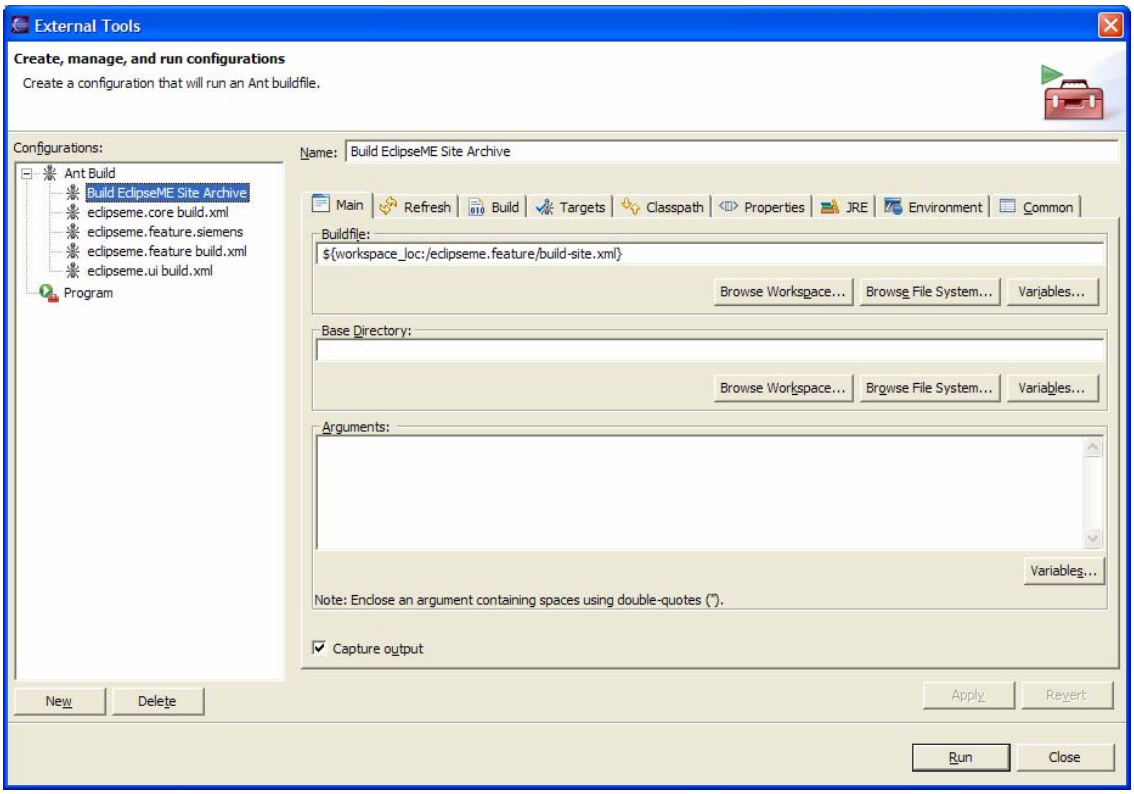
Building the EclipseME Distribution Archive

After development of any new functionality is complete, the EclipseME distribution archive can be built using the Plugin Development Environment facilities. The build process relies on the built-in Ant support in Eclipse. The steps required to build the distribution archive are as follows:

Within the eclipseme.feature project, locate the feature.xml file.

Right-click on feature.xml and choose Create Ant Build File from the PDE Tools submenu. This step will generate all the required build.xml files for each of the projects. Because of differences between developers' individual machine layouts, these files are generated, rather than being checked into the EclipseME CVS repository.

From the Eclipse Run menu, select External Tools and then External Tools... again from the submenu. This will display a dialog similar to the following:



Select the item labeled Build EclipseME Site Archive and press the Run button. This will start the process of packaging up the EclipseME distribution archive. By default, the output archive will be placed in a directory named build in the same parent directory as your workspace. Thus, if your Eclipse workspace is located in C:\eclipse\workspace then the archive will be built into the directory C:\eclipse\build.

Typically, once you have build the site archive once, Build EclipseME Site Archive will then appear directly in your Run / External Tools menu.

致谢
EclipseME 包含了下列公司和个人开发的软件
部分图标来自 Eclipse 项目
Copyright (c) 2000,2002 IBM Corporation 等
http://www.eclipse.org
Jetty HTTP Server
Copyright (c) Mort Bay Consulting Pty. Ltd. (Australia)等
http://jetty.mortbay.org
jawin
本产品包含了 DevelopMentor 开源项目开发的软件 (http://www.develop.com/OpenSource).
Yang Liu 为 J2ME classpath 容器的原始版本提供了主要技术支持。
Rafael Torres 提供了 Siemens 工具套件的最初技术支持.
同样感谢Lake Design, Inc(www.lakedesign.net)的Larry Gutman。 他设计了新版的 www.EclipseME.org。