在程序中实现对 java 源文件编译的 3 种方法

一般情况下对 java 源文件的编译均是在代码完成后使用 javac 编译的，不管是使用 IDE 还是直接使用命令行。这里要说的情况是比较特别的，就是在代码内动态的编译一些代码。比如你想通过在某个目录下通过放置一些源代码的方式来实现对程序功能的动态扩展，那么你的程序就需要具有一种对源代码的编译、加载、运行的能力，可能就需要本文介绍的 3 种方法。

方法 1：通过调用本机的 javac 命令来编译。

在 java 程序中调用 javac 命令可以通过调用 Runtime 类的 exec 或是 ProcessBuilder 类的 start 方法来完成，这两个类的功能基本相同，用法也比较相似，这里的例子我们就用 ProcessBuilder 来演示。如果是 JDK1.5 之前的版本请使用 Runtime 类完成相同的功能。

开始之前先来点准备工作，将下面的类放到 c:\mytest\src\ 目录下，这个类我们不会在 IDE 中编译，而是由我们程序完成其编译。保存时使用 UTF-8 格式。可以直接在附件中下载这个类。

Java 代 码

[img]http://devbbs.doit.com.cn/attachments/images/302-icon_copy.gif[/img]

```
public class HelloWorld {
    public void sayHello(String in)
    {
        System.out.println("动态编译成功");
        System.out.println("使用编译方式:" + in);
    }

}
```

```
public class HelloWorld {
    public void sayHello(String in)
    {
        System.out.println("动态编译成功");
        System.out.println("使用编译方式:" + in);
    }

}
```

准备工作完成，下面就看一下我们程序的代码，这里只列出主要代码

Java 代 码

[img]http://devbbs.doit.com.cn/attachments/images/302-icon_copy.gif[/img]

```
public class JavacCompile {
```

```
    private static String filePath = "c:\\mytest\\src\\HelloWorld.java";
    private static String binDir = "c:\\mytest\\bin";

    public static void main(String[] args) {
        File binOutDir = new File(binDir);
        if (!binOutDir.exists())
        {
            binOutDir.mkdirs();
        }

        // 设置 javac 的编译参数,使用-encoding 指定编码方式,-d 并指定编译生成 class
文件目录
        ProcessBuilder pb = new ProcessBuilder("javac","-encoding", "UTF-8","-d",
binDir, filePath);
        try {
            // 开始调用 javac 命令编译
            final Process proc = pb.start();

            // 处理进程的输出，避免挂死
            new Thread(new Runnable() {
                public void run() {
                    processStream(proc.getInputStream());
                    processStream(proc.getErrorStream());
                }
            }).start();

            // 等待编译完成
            proc.waitFor();

            // 加载编译好的类，并调用相应的方法
            new LoaderClassByDir(binDir).execSayHello("javac");

        } catch (Exception ex) {
            Logger.getLogger(JavacCompile.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }

    private static void processStream(InputStream stderr)  {
        ...
    }
}
```

```
public class JavacCompile {

    private static String filePath = "c:\\mytest\\src\\HelloWorld.java";
    private static String binDir = "c:\\mytest\\bin";

    public static void main(String[] args) {
        File binOutDir = new File(binDir);
        if (!binOutDir.exists())
        {
            binOutDir.mkdirs();
        }

        // 设置 javac 的编译参数,使用-encoding 指定编码方式,-d 并指定编译生成 class
文件目录
        ProcessBuilder pb = new ProcessBuilder("javac","-encoding", "UTF-8","-d",
binDir, filePath);
        try {
            // 开始调用 javac 命令编译
            final Process proc = pb.start();

            // 处理进程的输出，避免挂死
            new Thread(new Runnable() {
                public void run() {
                    processStream(proc.getInputStream());
                    processStream(proc.getErrorStream());
                }
            }).start();

            // 等待编译完成
            proc.waitFor();

            // 加载编译好的类，并调用相应的方法
            new LoaderClassByDir(binDir).execSayHello("javac");

        } catch (Exception ex) {
            Logger.getLogger(JavacCompile.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }

    private static void processStream(InputStream stderr)  {
        ...
    }
}
```

LoaderClassByDir 类的代码会保含在后面的上传的文件中，因为这里主要介绍完成程序中对 java 源文件的编译，对于类的加载和运行不多做描述，可以参考 LoaderClassByDir 类中的简单实现。

方法 2：使用 Sun 的 tools.jar 包时的 com.sun.tools.javac.Main 类完成对代码的编译。

注意这个类的是在 tools.jar 包里，tools.jar 不是标准的 Java 库，在使用时必须要设置这个 jar 的路径，使用 IDE 时需要显示的引入到编译路径中，不然会找不到。我们使用此类改写上面的编译类如下：

Java 代 码

[img]http://devbbs.doit.com.cn/attachments/images/302-icon_copy.gif[/img]

```
public class JavacCompile {

    private static String filePath = "c:\\mytest\\src\\HelloWorld.java";
    private static String binDir = "c:\\mytest\\bin";

    public static void main(String[] args) {
        File binOutDir = new File(binDir);
        if (!binOutDir.exists())
        {
            binOutDir.mkdirs();
        }

        // 将编译参数通过数组传递到编译方法中,该函数的方法和 javac 的参数完成一致
        Main.compile(new String[]{"-encoding", "UTF-8","-d", binDir, filePath});

        try {
            // 加载编译好的类，并调用相应的方法
            new LoaderClassByDir(binDir).execSayHello("sun tools");
        } catch (Exception ex) {
            Logger.getLogger(JavacCompile.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }
}

public class JavacCompile {

    private static String filePath = "c:\\mytest\\src\\HelloWorld.java";
    private static String binDir = "c:\\mytest\\bin";
```

```
public static void main(String[] args) {
    File binOutDir = new File(binDir);
    if (!binOutDir.exists())
    {
        binOutDir.mkdirs();
    }

    // 将编译参数通过数组传递到编译方法中,该函数的方法和 javac 的参数完成一致
    Main.compile(new String[]{"-encoding", "UTF-8","-d", binDir, filePath});

    try {
        // 加载编译好的类，并调用相应的方法
            new LoaderClassByDir(binDir).execSayHello("sun tools");
        } catch (Exception ex) {
            Logger.getLogger(JavacCompile.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }
}
```

使用这个类后，同样的功能代码变得更加简洁。

方法 3：使用 javax.tools 包

从上面可以看到方法 2 的缺点就是 tools.jar 需要我们自行导入。而在 Java SE6 中为我们提供了标准的包来操作 Java 编译器，这就是 javax.tools 包。使用这个包，我们可以不用将 jar 文件路径添加到 classpath 中了。 使用这个类的方法和上面的类很相似，我只需要将

Java 代 码

[img]http://devbbs.doit.com.cn/attachments/images/302-icon_copy.gif[/img]

```
Main.compile(new String[]{"-encoding", "UTF-8","-d", binDir, filePath});

Main.compile(new String[]{"-encoding", "UTF-8","-d", binDir, filePath});
```

替换成：

Java 代 码

[img]http://devbbs.doit.com.cn/attachments/images/302-icon_copy.gif[/img]

```
// 将编译参数通过数组传递到编译方法中，该函数的方法和 javac 的参数完成一致
        JavaCompiler compiler = ToolProvider.getSystemJavaCompiler();
        compiler.run(null,  null,  null,  "-encoding",  "UTF-8","-d",  binDir,
filePath);
```

```
// 将编译参数通过数组传递到编译方法中，该函数的方法和 javac 的参数完成一致
        JavaCompiler compiler = ToolProvider.getSystemJavaCompiler();
        compiler.run(null,  null,  null,  "-encoding",  "UTF-8","-d",  binDir,
filePath);
```

就可以完成相应的编译功能，这里简介一下 run 的使用方法：

注意：使用上传文件中的代码做测试时，为避免上次编译的影响记得手动删除 C:\mytest\bin 下的类文件