Working with files and directories in Java

by Keld H. Hansen

Java contains a lot of useful utility packages. One is java.util with its lists, maps, and calendar stuff--a lot to look into and use in your applications. Another package, java.io, contains what you need for reading and writing files and handling data streams in general. In this article I'll give examples on the use of java.io for constructing a directory tree.

To spice it up I have added a spoonful of HTML, a dash of recursive methods calls, and a pinch of XML. Hope it tickles those refined taste buds!

The File class from the java.io package

We find the tools for getting information about files and directories in Java's File class (in the java.io package). First we create an instance of the class by giving the name of the file (or directory). There are several constructors, but the simplest is this (using Windows file-syntax):

```
File f = new File
        ("c:\\server\\classes\\hansen\\playground\\MyFile.class");
```

You may also use a relative file name like this:

```
File f = new File("hansen\\playground\\MyFile.class");
```

To use this format you'll have to know the "current user directory", which most often is where the Java Virtual Machine is invoked. If you are in doubt you can get the name of the current user directory like this:

```
String userdir = System.getProperty("user.dir");
```

Several methods are available for inspecting an instance of the File class. Some of the important ones are:

Method	Purpose
boolean exists()	does the file exist?
boolean isFile()	is it a file?
boolean isDirectory()	… or a directory?
String[] list()	return the names of all files and directories in a directory
String getName()	get the file or directory's name
String getPath()	get the file or directory's path

These methods are actually all we need in order to find all files and directories in a given directory--all the way down to the last leaves in the directory tree. Let's sketch some pseudo-code for an algorithm:

```
directory-process:
for every item in this directory
  if item is a file then display its name
  if item is a directory then
    display its name
    repeat directory-process for item
  end-if
end-for
```

Simple, isn't it? And as you can see--we get a chance to use a recursive method-call--which most programmers find exciting, since it simplifies the code and makes it more readable.

The MyFile class

Simply listing the contents of the directory does not give us much flexibility in our program, so we'll start by building a matching directory tree structure in memory. For this purpose we need two simple classes: one for a file and one for a directory. The file class is very simple:

```
package hansen.playground;

public class MyFile {
  private String path;
  private String name;

  /*
  * Set path and file name.
  * Example: For the file "MyFile.class" in
  * "classes\hansen\playground" we have:
  *   file = "MyFile.class" and
  *   path = "classes\hansen\playground"
  */
  public MyFile(String path, String name) {
    this.path = path;
    this.name = name;
  }

  /*
  * Get the name of the file
  */
  public String getName() { return name; }

  /*
  * Get the path of the file
  */
  public String getPath() { return path; }
}
```

In order to facilitate coding we have split the name of a file in a "path" part and a "file name" part.