Software Engineer

How do I... Recursively scan directories with PHP's DirectoryIterators?

By Melonfire
June 15, 2007, 8:00 AM PDT

This blog entry is also available as a TechRepublic download, which contains all of the sample code in a manageable text file.

One of PHP5's most interesting new features is the addition of Iterators, a collection of ready-made interfaces designed to help in navigating and processing hierarchical data structures. These Iterators significantly reduce the amount of code required to process an XML document tree or a file collection. A number of Iterators are available, including the *ArrayIterator*, *CachingIterator*, *LimitIterator*, *RecursiveIterator*, *SimpleXMLIterator* and *DirectoryIterator*.

It's this last Iterator that's the subject of this How do I... tutorial. The *DirectoryIterator* provides a quick and efficient way of processing the files in a directory; with a little creative coding, it can also be used to recursively process a nested directory tree. Both these tasks can be accomplished using just a few lines of code, representing a significant improvement over the "standard" way of doing things.

Processing a single-level directory

Let's begin with something simple: processing a single-level directory. Type (or copy) the following script (**Listing A**), altering the directory path to reflect your local configuration:

Listing A

```
<?php
$it = new DirectoryIterator("/tmp/mystuff");
foreach($it as $file) {
if (!$it->isDot()) {
echo $file . "\n";
}
}
?>
```

When you view the output of this script in your browser, you should see a list of the files in the named directory. How did this happen? Well, the *DirectoryIterator* class provides a pre-built interface to iterating over the contents of a directory; once instantiated with the location of the target directory, it can then be processed as though it were a standard PHP array, with each element representing a file in the directory. Note the use of the *isDot()* method to filter out the "." and ".." directories, respectively.

Processing a nested directory tree

Recursively processing a nested directory tree is almost as simple. In this case, the *DirectoryIterator* needs to check each object it encounters within the first-level directory, determine whether it is a file or directory, and, if a directory, drill one level deeper to examine the next level of contents. This sounds fairly complex, and in the past could easily add up to 15-plus lines of code.

With PHP5, though, all you need are two new Iterators: the *RecursiveDirectoryIterator* and the *RecursiveIteratorIterator*, which together incorporate all the above functionality. Take a look at **Listing B**:

Listing B

```
<?php
$it = new RecursiveDirectoryIterator("/tmp");
foreach(new RecursiveIteratorIterator($it) as $file) {
echo $file . "\n";
}
?>
```

In this case, the output should now include a list of all the files and directories under the starting directory. Needless to say, this kind of built-in recursive interface is very handy for situations that require you to process all the files under a particular directory level — for example, when recursively compressing a directory tree, or altering group/owner permissions on a series of nested files.

A real-world application: Printing a directory tree

A common application of directory recursion involves printing a graphical directory tree. With Iterators, this task is a snap, because included within the Iterator class documentation is an example class written specifically for this purpose. The *DirectoryTreeIterator* (credit: Marcus Boerger) provides additional enhancements to the *RecursiveIteratorIterator* discussed previously, most notably ASCII markers that represent depth and location within the tree structure.

You can examine the source code for this example class on the php.net Web site.

Listing C shows how the *DirectoryTreeIterator* can be used.

Listing C

```
<?php
$it = new DirectoryTreeIterator("/tmp/cookbook/");
foreach($it as $path) {
echo $path . "\n";
}
?>
```

And here's a brief snippet of the output you might see:

```
|-ch01
| |-recipe01
| | |-example01.php
| | \-example02.php
| |-recipe02
| | |-example01.php
| | \-example02.php
| |-recipe03
| | \-example01.php
...
```

To better understand the value-add of these various *DirectoryIterators*, try coding the three applications demonstrated in this tutorial using standard file and directory functions. Once you're done, you'll have a new appreciation for the simplicity and ease of use the *DirectoryIterators* bring to PHP5. Happy coding!