

## TYPO3.Flow - Task #51530

### Improve speed of Files::readDirectoryRecursively using RecursiveDirectoryIterator?

2013-08-29 16:23 - Jacob Floyd

<b>Status:</b>	New	<b>Start date:</b>	2013-08-29
<b>Priority:</b>	Should have	<b>Due date:</b>	
<b>Assignee:</b>		<b>% Done:</b>	0%
<b>Category:</b>		<b>Estimated time:</b>	0.00 hour
<b>Target version:</b>		<b>Has patch:</b>	No
<b>Sprint:</b>		<b>Complexity:</b>	
<b>PHP Version:</b>			

#### Description

#### Basics (TL;DR)

TYPO3/Flow/Utility/Files::readDirectoryRecursively uses \DirectoryIterator instead of \RecursiveDirectoryIterator. I suspect that an implementation using \RecursiveDirectoryIterator would be faster.

Let's write a version of readDirectoryRecursively that uses \RecursiveDirectoryIterator and profile the two versions to see which one is faster.

#### Background

When I asked about it in IRC, ChristianM said this:

ChristianM: if I remember right there was some oddness to the RecursiveDirectoryIterator that we wanted to avoid but I don't remember the specifics

Consider this a selection of [PHP Release dates](#)

```
01 May 2008 - PHP 5.2.6
10 May 2008 - Flow's Files::readDirectoryRecursively introduced
30 Jun 2009 - PHP 5.3.0
17 Sep 2009 - PHP 5.2.11
19 Nov 2009 - PHP 5.3.1
```

According to the [Changelog for \RecursiveDirectoryIterator](#)

```
5.2.11 - Introduced RecursiveDirectoryIterator::FOLLOW_SYMLINKS
5.3.0 - Extends FilesystemIterator (which extends DirectoryIterator) instead of DirectoryIterator
5.3.0 - Implements SeekableIterator
5.3.1 - Introduced RecursiveDirectoryIterator::FOLLOW_SYMLINKS
```

We don't even support 5.2 anymore, and haven't for a long time. I suspect that the oddness with \RecursiveDirectoryIterator has been fixed already, so I think that using \RecursiveDirectoryIterator is probably feasible now.

#### Stub Method

This is the same function only it doesn't filter hidden directories, just hidden files.

```
static public function readDirectoryRecursively($path, $suffix = NULL, $returnRealPath = FALSE
, $returnDotFiles = FALSE, &$filenames = array()) {
    if (!is_dir($path)) {
        throw new Exception("'" . $path . "' is no directory.", 1207253462);
    }

    $directoryIterator = new \RecursiveIteratorIterator(
```

```

        new \RecursiveDirectoryIterator(
            $path,
            \FileSystemIterator::UNIX_PATHS|\FileSystemIterator::SKIP_DOTS|\FileSystemIterator
::FOLLOW_SYMLINKS
        )//,
        //\RecursiveIteratorIterator::SELF_FIRST //returns directories as well.
    );
    $suffixLength = strlen($suffix);

    foreach ($directoryIterator as $pathname => $fileInfo) {
        $filename = $fileInfo->getFilename();
        if ($returnDotFiles === FALSE && $filename[0] === '.') {
            continue;
        }
        //This is needed if directories are included (see SELF_FIRST above)
        //if ($fileInfo->isFile() && ($suffix === NULL || substr($filename, -$suffixLength) ==
= $suffix)) {
            if ($suffix === NULL || substr($filename, -$suffixLength) === $suffix) {
                $filenames[] = self::getUnixStylePath(($returnRealPath === TRUE ? realpath($pathna
me) : $pathname));
            }
        }
    }
    return $filenames;
}

```

## History

### #1 - 2013-08-29 17:55 - Jacob Floyd

OK. This one is functionally equivalent including removing hidden directories. However, it uses \RecursiveCallbackFilterIterator which is only available in PHP 5.4.

```

    static public function readDirectoryRecursively($path, $suffix = NULL, $returnRealPath = FALSE, $returnDot
Files = FALSE) {
        if (!is_dir($path)) {
            throw new Exception("'" . $path . "' is no directory.", 1207253462);
        }

        $suffixLength = strlen($suffix);
        $filter = function ($fileInfo, $pathname, $iterator) use ($suffix, $suffixLength, $returnDotFiles) {
            $filename = $fileInfo->getFilename();
            if ($returnDotFiles === FALSE && $filename[0] === '.') {
                return FALSE;
            }
            if (($fileInfo->isFile() && ($suffix === NULL || substr($filename, -$suffixLength) === $suffix)) |
| $iterator->hasChildren()) {
                return TRUE;
            }
            return FALSE;
        };

        $directoryIterator = new \RecursiveIteratorIterator(
            new \RecursiveCallbackFilterIterator(
                new \RecursiveDirectoryIterator(
                    $path,
                    //defaults: \FileSystemIterator::KEY_AS_PATHNAME|\FileSystemIterator::CURRENT_AS_FILEINFO
                    \FileSystemIterator::UNIX_PATHS|\FileSystemIterator::SKIP_DOTS|\FileSystemIterator::FOLLOW
_SYMLINKS
                ), $filter
            )
        );

        foreach ($directoryIterator as $pathname => $fileInfo) {
            $filenames[] = self::getUnixStylePath(($returnRealPath === TRUE ? realpath($pathname) : $pathname)
);
        }
        return $filenames;
    }

```

### #2 - 2013-08-29 19:49 - Jacob Floyd

I've been working on this, because I needed a more robust filtering mechanism when using readDirectoriesRecursively.

You can see my final version, based on the one in the previous comment [here](#)