

Using SPL Tools in Your Code



OTHERWISE KNOWN AS SPL IN THE WILD



If you don't know what SPL is or have never used SPL features, this talk might go fast for you.

Feedback is good!





Why SPL?
What SPL?
How SPL?

Standard **PHP** Library

A library of standard interfaces, classes, and functions designed to solve common programming problems and allow engine overloading

But isn't SPL an Extension?



- SPL is an extension
- SPL is a core extension
- SPL cannot be built shared
- SPL should not be turned off
- SPL is present in PHP since 5.0 (almost 6 years ago)
- As of 5.3, SPL cannot be turned off

If you don't have SPL, whoever built your PHP is an idiot.

Autoload



DON'T LET FRIENDS USE `__AUTOLOAD`

Autoload Stack err Queue



- `spl_autoload_register()`
- `spl_autoload_unregister()`
- `spl_autoload_call()`
- `spl_autoload_functions()`

- <https://github.com/hyla/hyla/blob/a9f445115b942865241653dce02aa029c7453f41/kohana/application/bootstrap.php>

Beware the Weirdness



- `spl_autoload()`
 - lowercase
 - relative paths
 - namespace aware
- `spl_autoload_extensions()`
 - include the `.`
 - no spaces in between
 - comma separated string

Exceptions



WHAT WENT WRONG WHERE

Exception Classes



LogicException

BadFunctionCall
Exception

Domain
Exception

InvalidArgument
Exception

Length
Exception

OutOfRangeException

BadMethodCall
Exception

Exception Classes



RuntimeException

OutOfBounds
Exception

Overflow
Exception

Range
Exception

Underflow
Exception

Unexpected
Value
Exception

Marker Interface



- Note that the ability to “nest” exceptions in 5.3 makes using this stuff far more practical
- Component level interface as the “marker” or “domain” for all exceptions belonging to a library or package
- Specific types of exceptions for code that tell what happened

Arrays as Objects



**I'M AN ARRAY, WELL KIND OF BUT
I'M AN OBJECT, REALLY!
I AM INCREDIBLY USEFUL**

The use cases are infinite



- <https://github.com/ooyes/Microweber/blob/d4d64cb563b73fa47db5e1e61f913d426d29154c/system/application/stats/libs/Zend/Registry.php>
- **Gotchas:**
 - &offsetGet
 - Quite buggy in 5.2, much more stable in 5.3
 - Can't use all array functions
 - Serialize, unserialize, sleep, wakeup issues

ArrayAccess, ArrayObject, Array



- Lots of people want arrayaccess or arrayobject to work like an array
- ArrayObject might be feasible and in fact some of these can already be used
- What about sorting? Merging? Other behavior

Files



**YES PHP HAS A FILE OBJECT
WELL ACTUALLY IT HAS 3
SPLFILEINFO
SPLFILEOBJECT
SPLTEMPFILEOBJECT**

Practical Uses



- <https://github.com/fqqdk/phpcmc/blob/c3b2450ff6f0293f3de0525745f4f5d551181dod/src/jabbar/PearXmlBuilder.php>
- <https://github.com/horde/horde/blob/a49d5c901d184dd8581ada0829ad33db053fd07b/framework/Support/lib/Horde/Support/StringStream.php>

Interfaces



MAGIC AND ORGANIZATION

PHP Interfaces



- Traversable
- Iterator
- IteratorAggregate
- ArrayAccess
- Serializable

- And in 5.4 JsonSerializer

Countable



- Interface meaning “you can count me”
- Can be put on any class
- Makes `count()` magical
- Note this is NOT the same as `iterator_count()`

Iterator Interfaces



- Outer Iterator
 - Inner Iterator
 - Seekable Iterator
-
- For examples, pay attention to what all the iterator classes we'll talk about implement

SplSubject SplObserver



- Are you implementing the observer pattern in your code?
- Do you intend to have other people use your code/library in some way?
- Are you implementing something LIKE the observer pattern?
- <http://www.a-scripts.com/object-oriented-php/2009/02/21/the-observer-pattern/>

Iterators



TAKE A DRINK ...

Why Iterators?



- Readable
- Easy to understand
- Testable
- Easy to refactor
- Consistent usage
- Easily add functionality
- Can be extended
- Can be chained

Meet the Iterator Interface



Start Page

iterator.php ×

```
<?php
```

```
interface Iterator extends Traversable {  
    abstract public current();  
    abstract public key();  
    abstract public next();  
    abstract public rewind();  
    abstract public valid();  
}
```

So how is it different?



\$ar = array();

- *reset(\$ar)*
- *!is_null(key(\$ar))*
- *current(\$ar)*
- *key(\$ar)*
- *next(\$ar)*

\$it = new Iterator;

- *\$it->rewind()*
- *\$it->valid()*
- *\$it->key()*
- *\$it->current()*
- *\$it->next()*

FilterIterator



- Abstract Class
- Has one method that must be implemented – accept – which should return true or false
- Highly useful for many types of iteration
- <https://github.com/sebastianbergmann/php-file-iterator/blob/master/File/Iterator.php>



IteratorIterator



- Regular Class
- Stick in something that implements traversable
- Instant Iterator
- <https://github.com/mediaslave/phrails/blob/6d40d1584c0202668b071cofd4b53e060b6724c7/framework/db/ResultSet.php>



ArrayIterator



- Regular Class
- Iterates an array – OR the public properties of an object! (neat trick – dirty trick)
- <https://github.com/codealchemy/ACL/blob/d1729ec7c3aee427fb57da9c2f2817503aa507b5/lib/CODEAlchemy/Component/ACL/NodeList.php>



RecursiveIteratorIterator



- Regular Class
- Like IteratorIterator only recursive to boot
- <https://github.com/Xosrofox/propelVendor/blob/85986d1ef52167f3044d69be142c72895174bac1/propel/config/PropelConfigurationIterator.php>



ParentIterator



- Regular Class
- Filter out stuff without children
- <https://github.com/ebihara/php-src/blob/b62203bf12ea1c2b63a0de996a236f43b2ac100c/ext/phar/phar/directorygraphiterator.inc>



LimitIterator



- Regular Class
- Like mysql's limit – pick your range and offset and foreach away!
- <https://github.com/mintao/zf-mini/blob/9efff5f8b25383ef2cdf91b0e4301d90965c4fb8/Paginator/SerializableLimitIterator.php>



CachingIterator



- Regular Class
- Manages another iterator by checking whether it has more elements each time using a hasNext() method
- https://github.com/shimondoodkin/node_spreadsheet/blob/e16660cebd2b65abba1a8f5db4449285fa4c3cc6/orig/Classes/PHPExcel/Worksheet/RowIterator.php



RecursiveCachingIterator



- Regular Class
- Just like caching iterator only – believe it or not – recursive!
- https://github.com/codedor/cakephp_navigation/blob/121b3f0594adoe42a873f65d45583c5ca85343c3/views/helpers/menu.php



DirectoryIterator



- Regular Class
- Makes going through directories a snap
- isDot, isFile – I love you
- <https://github.com/quentinhill/curator/blob/44ec76ca9c3d749028b5d950ab78c7dc480dee83/Curator/Project.php>



RecursiveDirectoryIterator



- Regular Class
- Like, do it again... and again... and again... and...
- <https://github.com/gjarkko/Track-o-Matic/blob/ce506e71326805fe24e38b4262328648cad9a50e/server/classes/Tools.php>



RegexIterator



- Regular Class
- Filter an iterator by a regex
- Pick how you want it to match
- <https://github.com/drm/Sprig/blob/b57bee029ce3d4317f90f38de2b643e3b6d94c96/lib/Sprig/Extension/Smarty/PluginLoader/Iterator.php>



Datastructures



NEW WAYS OF MANAGING DATA

<http://matthewturland.com/2010/05/20/new-spl-features-in-php-5-3/>

DoublyLinkedLists – CS Lesson



- ordered collection of values
 - linked to each element before it
 - linked to each element after it
 - “doubly linked”
-
- PHP datastructure – a php object with a doublylinkedlist stored inside it

SplDoublyLinkedList



- Don't use this
- Yes, that's a terrible thing to say – but this is really nothing more than a “base class” with little to recommend on its own
- Has a doublylinkedlist from C underneath instead of a hashtable – if you know what that means you may find a real use for this (I have not)

SplStack



- Data is in LIFO
- Anything you need to iterate a lot
- Even cooler? Turn on the mode that will autodelete each item as you process it
- Any Queue you need to push stuff onto and deal with in LIFO order
- <https://github.com/CHH/Spark2/blob/a01607a2480bd6a9b8251e56579fe84869683925/lib/Spark/App.php>

SplQueue



- Data is in FIFO
- Anything you need to iterate a lot
- Even cooler? Turn on the mode that will autodelete each item as you process it
- Any Queue you need to push stuff onto and deal with in LIFO order
- <https://github.com/composer/composer/blob/52565a593529048a631110879c76b46bbf76ba82/src/Composer/DependencyResolver/Solver.php>

Heap – CS Lesson



- comparison function used to compare the new element to other elements
- element is placed according to functions return value
- underlying algorithm does it with minimal comparisons

- PHP datastructure – a php object with a heap stored inside it

SplMinHeap, SplMaxHeap



- These are concrete heap implementations, designed to grab the lowest possible value out, or the highest possible
- <https://github.com/Wordi/LooPHP/blob/fce3947bc27cdd4ae2109e62f7ad58ed68af7862/LooPHP/EventLoop.php>

SplPriorityQueue



- Uses heap internally
- Is non-deterministic when identical priorities are used (bug bug bug....)
- <https://github.com/ss23/DeBot/blob/dced95faaf5b5284057d0824151bae4a95866305/core/SplFIFOPriorityQueue.php>

SplFixedArray



- You have a large amount of data, you know the final size, you need to stick it into an array
- You're not going to expand it past the final size
- This is not a small amount of data
- You might need it in non-sequential order but can handle having only integers for keys
- <https://github.com/cboden/gClient/blob/429b27a54db5b4c1d30bbdc83566678c03b96bea/lib/gClient/Calendar/Calendar.php>

SplObjectStorage



- This can be used two ways
 - Objects can be keys in an array (with two values)
 - As a Set (with one value)

- https://github.com/greggles/epm_project_management/blob/50b904333647216ec678fd922f4ac53dbfoe53ee/QueryPath/CssEventHandler.php

What do you want to see in SPL?



- More standard interface?
- More datastructures?
 - trees?
 - graphs?
- More iterators? really? more?

Get Involved



- <http://edit.php.net>
- Blog posts
- Articles
- Use SPL in the wild

Contact Me



- <http://emsmith.net>
- <http://joind.in/3791>
- auroraeosrose@gmail.com
- IRC – freenode – auroraeosrose
- #php-gtk #coapp and others