international PHP2007 conference

Enhanced StreamWrappers

Frank Kleine, 1&1 Internet Inc.

About me

- Frank Kleine
- Working at 1&1 Internet, Inc
- PHP since 2000 (Java since 2006)
- Lead Developer Stubbles
- Lead Developer XJConf for PHP
- Co-Author „Exploring PHP“

Agenda

- What are StreamWrappers?
- Your own StreamWrapper
- Files and more
- Ease your XML with XInclude
- Phar, star and xar
- Going mad: AOP at runtime
- Q&A

What are StreamWrappers?

- Stream: sequence of data
- Wrapper: how to handle a stream
- PHP StreamWrapper: since 4.3.0 for file system, sockets, HTTP[S], input & output, compression and more
- All file functions in PHP use streams and therefore stream wrapper
- Possibility to write and register your own StreamWrapper

Your own StreamWrapper

```
class MyStream  {
    stream_open($path, $mode, $options, $opened_path)
  { ... }
    stream_close() { ... }
    stream_read($count) { ... }
    stream_write($data) { ... }
    ....
}
stream_wrapper_register('myStream', 'MyStream');
```

Your own StreamWrapper 2

- Usage:

```
$fp   = fopen('myStream://myUrl');
$data = fgets($fp);
fclose($fp);
```

- Consult PHP manual for a complete list of methods: http://php.net/stream-wrapper-register

Files and more

- Simplest usage of a StreamWrapper: replace or extend pathes

```
class TranslatorStream {
  stream_open($path, ...) {
    $realPath = str_replace('%lang%',
  getUsedLanguage(), $path);
    // now open $realPath somewhere on disc
  }
  // other stream methods
}
stream_wrapper_register('trans', 'TranslatorStream');
$txt =
  file_get_contents('trans://txt/%lang%/main.txt');
```

Files and more, cont.

- This method can be unit tested:

```
class Foo {
  public function read($file) {
     $data = file_get_contents($file);
     // do something with data
     return $data;
  }
}
```

- Without having real files on the disc:

```
assertEqual($foo->read('test://testdir/file.txt',
   'processedData');
```

- Just create a stream wrapper for test://

Files and more, cont. 2

- Currently in development as SimpleTest extension:

```
SimpleTestStreamWrapper::register();
SimpleTestStreamWrapper::setRoot(new
                SimpleTestStreamDirectory('testdir'));
$file = new SimpleTestStreamFile('file.txt');
$file->setContent('data to process');
SimpleTestStreamWrapper::getRoot()->addChild($file);
```

- Gives the following virtual file system:

```
/testdir
    /file.txt
```

Ease your XML with XInclude

- XML allows to include other XML files into the document

```
<?xml version="1.0" encoding="iso-8859-1"?>
<root xmlns:xi="http://www.w3.org/2001/XInclude">
  <aTag>
    <xi:include href="other.xml"/>
  </aTag>
</root>
```

- PHP has XInclude support:
  `DOMDocument->xinclude();`

Ease your XML with XInclude, cont.

- PHP's XInclude support can be used with stream wrappers as well:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<root xmlns:xi="http://www.w3.org/2001/XInclude">
  <aTag>
    <xi:include href="createXMLOnTheFly://foo.bar"/>
  </aTag>
</root>
```

- allows to create arbitrary XML on the fly depending on other conditions

Ease your XML with XInclude, cont. 2

- concrete usage: Stubbles XML/XSL view engine
- include other XML files dynamically depending on external parameters
- apply a XSL stylesheet first before stream wrapper returns XML data
- results in a kind of recursive applyment of the XSL stylesheet and a totally dynamical created XML file

Ease your XML with XInclude, cont. 3

- Not just XInclude, XSL import is supported as well:

```
<xsl:stylesheet version="1.1"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:import href="dynamicXSL://copy.xsl"/>
  ...
</xsl:stylesheet>
```

- Gives you the whole power of PHP just inside of XML and XSL files!

Phar, star and xar

- All PHP files in one big file (similar to .jar in the Java world)
- Just need to ship one file to the customer (almost; images, CSS and JavaScript files stay seperate)
- Even self-running versions available
- Phar: (semi-)official PHP version
- Star, xar: inspired by phar but independent developments from Stubbles and XP-Framework developers

Phar, star and xar, cont.

- You can create your own all-in-one-file-format, too! (Not that you should...)
- Its all about stream wrappers:
  `include 'star://net.stubbles.package.OneClass';`
- star:// redirects PHP internal code to your stream wrapper (if registered before)
- stream_open() will receive `'star://net.stubbles.package.OneClass'` as argument, its up to the stream wrapper to make something useful out of this and to return the correct PHP code

Phar, star and xar, cont. 2

- Self running versions: place plain PHP code at beginning of file that is required to read contents from this file
- `__halt_compiler()` stops PHP to continue parsing, compiling and executing the file at this point, after that point any binary data can be put into the file
- `__COMPILER_HALT_OFFSET__` marks position of binary data start

Phar, star and xar, cont. 3

```
<?php
// some php code here
// and some code to read the virtual php files
// from this file
__halt_compiler();?>this represents any binary data
  and will not be parsed by PHP
...
```

Final tip: when reading from such a file: fopen() only once, on any subsequent reads just fseek()

While loading the code…

… why not change it inbetween?

- We read code from the file and hold it in a variable before returning it from the stream wrapper.
- Wouldn't this be an ideal place to change the code before it gets parsed, compiled and executed?

AOP at runtime

- AOP: Aspect-Oriented Programming
- Separation of concerns: especially crosscutting concerns
- Normally part of other methods (logging, security, etc.), e.g. mixed with business logic
- In AOP: seperate them into Aspects
- Aspects alter the behavior of the base code by applying additional behavior at various points in a program

AOP at runtime, cont.

- Several solutions for PHP:
  - phpAspect: Weaving source code before being put into production
  - aoPHP: PHP extension offering additional instructions within the language
  - S2Container.PHP5: configured with XML files
- But they require additional code for building or additional non-PHP-knowledge.

AOP at runtime, cont. 2

- Loading PHP code via a stream wrapper: why not modify the code in the moment it is loaded?
- Yeah, I know, performance...
- But it is possible!

AOP at runtime, cont. 3

- Two additional steps:
  - load PHP code in stream wrapper into a string
  - parse code, e.g. with `token_get_all()`
  - add Aspects wherever you like
  - return changed PHP code
- Example implementation in XP-Framework, but rarely usable outside of this framework

AOP at runtime, cont. 4

- So, it is possible, but
  - performance?
  - reliability?
  - usability?
  - does the code really do what it says?
- More research needed, but that requires more stream wrapper implementations to spread the news.

Lessons learned

- Stream wrappers are one of the most powerful tools PHP offers
- Lot of usage scenarios
- Use stream wrappers to simplify your life
- Do not overuse them
- Spread the news: every interested PHP developer should have implemented a stream wrapper

Thank you!

- Got bored? Sorry.

```
if (is_bullshit()) {
    throw away();
}
```

- Questions?

Links

- Slides: http://talks.frankkleine.de/
- Stubbles: http://stubbles.net/
- Phar in the PHP manual: http://php.net/phar