Streams, Sockets and Filters Oh My!

using and understanding PHP streams

Abstracting IO - Streams

Definitions

- Idea originating in 1950's
- Standard way to get Input and Output
- A source or sink of data

Who uses them

- C – stdin, stderr, stdout
- C++ iostream
- Perl IO
- Python io
- Java
- C#

TERMINOLOGY

Είναι πολύ σημαντικό να κατανοήσουμε τους όρους ότι το εγχειρίδιο χρησιμοποιεί για να εξηγήσει πώς PHP κάνει ρέματα.

And NOT in Greek:

It is very important to understand the terms that the manual uses to explain how PHP does streams.

- Stream
- Socket
- Filter

- Transport
- Wrapper
- Context

- Scheme
- Target

Definitions

- Stream
  - Resource that exhibits a flow or sucession of data
- Socket
  - Bidirectional network stream that speaks a protocol
- Filter
  - Performs operations on data as it is read from or written to a stream

Definitions

- Transport
  - Tells a network stream how to communicate
- Wrapper
  - Tells a stream how to handle specific protocols and encodings
- Context
  - A set of parameters and options to tell a stream (or socket or filter) how to behave

Definitions

- Scheme
  - *The* name of the wrapper to be used. file, http, https, ftp, etc.
- Target
  - Depends on the wrapper, filesystem uses a string path name, ssh2 uses a PHP resource

home/bar/foo.txt
file:///home/bar/foo.txt
http://www.example.com/foo.txt
ftp://user:pass@ftp.example.com/foo.txt

php://filter/read=string.toupper|string.rot13/resource=http://www.example.com

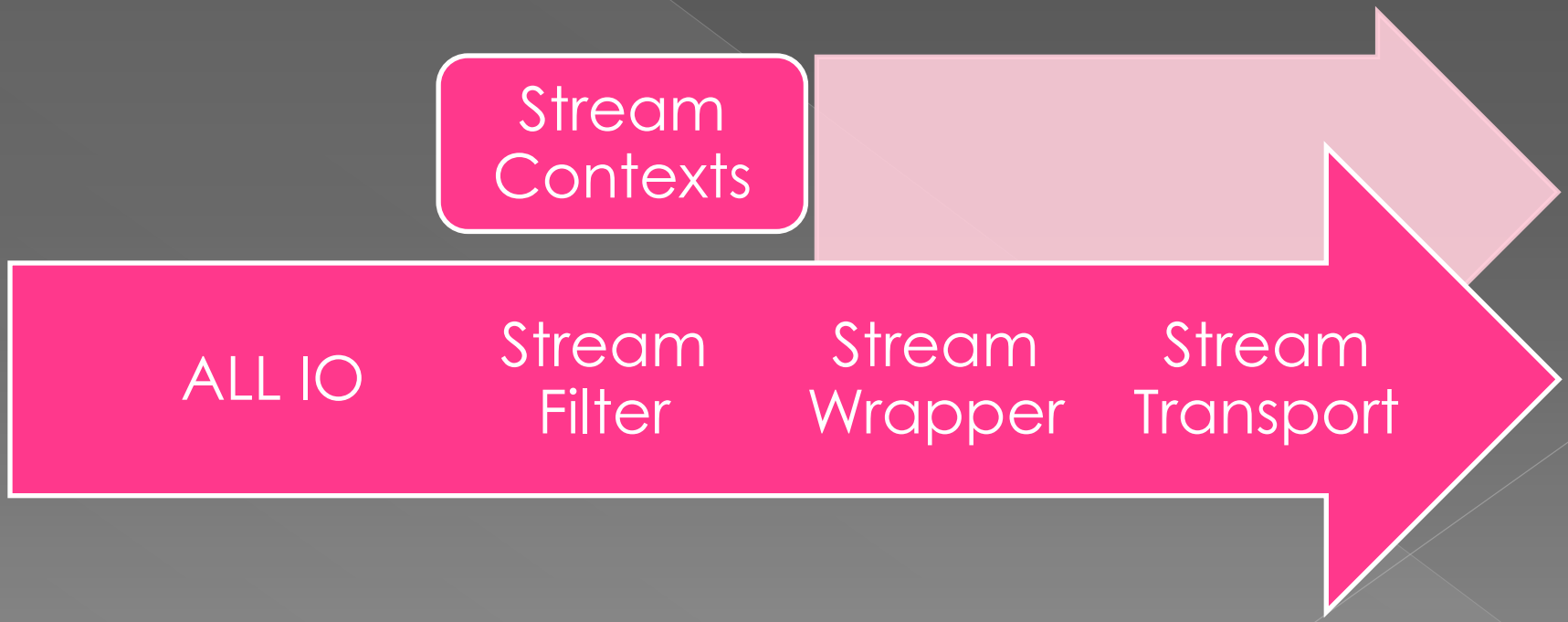
What uses streams?

- EVERYTHING
- include/require _once
- stream functions
- file system functions
- many other extensions

The Basics

What are and how to use
Streams, Sockets and Filters

How PHP Streams Work



What is a Stream?

- Access input and output generically
- Can write and read linearly
- May or may not be seekable
- Comes in chunks of data

Using Streams

```php
<?php
// best if you need the whole file
$data = file_get_contents('/home/foo/bar.txt');
// best to limit memory consumption and do chunks
// 8192 is internal chunk size
$fp = fopen('/home/foo/bar.txt', 'r');
while(!feof($fp)) {
        $data .= fread($fp, 8192);
}
fclose($fp);
// most memory efficient but slowest
// gets only a line at a time
$fp = fopen('/home/foo/bar.txt', 'r');
while(!feof($fp)) {
        $data .= fgets($fp);
}
fclose($fp);
```

Things to watch for!

- flock
- transport and wrapper limitations
- non-existent pointers (infinite loops can and will happen)
- error handling

What are Filters?

- Performs operations on stream data
- Can be prepended or appended (even on the fly)
- Can be attached to read or write
- When a filter is added for read and write, two instances of the filter are created.

Using Filters

```php
<?php
$html = 'Bad and <b>Naughty</b> tags are <i>here</i>.';

$fp = fopen('php://output', 'r');
stream_filter_prepend($fp, 'string.strip_tags');
fwrite($fp, $html);
fclose($fp);
```

Things to watch for!

- Data has an input and output state
- When reading in chunks, you may need to cache in between reads to make filters useful
- Use the right tool for the job

What are Sockets?

- Network Stream, Network Transport, Socket Transport
- Slightly different behavior from a file stream
- Bi-directional data

Using Sockets

```php
<?php
$fp = fsockopen("www.example.com", 80, $errno, $errstr, 30);
if (!$fp) {
    echo "$errstr ($errno)<br />\n";
} else {
    $out = "GET / HTTP/1.1\r\n";
    $out .= "Host: www.example.com\r\n";
    $out .= "Connection: Close\r\n\r\n";
    fwrite($fp, $out);
    while (!feof($fp)) {
        echo fgets($fp, 128);
    }
    fclose($fp);
}
```

Things to watch for!

- Sockets block
  - stream_set_blocking
  - stream_set_timeout
  - stream_select
- feof means “connection_closed”?
- huge reads or writes (think 8K)
- stream_get_meta_data is READ ONLY

That sockets extension…

- New APIS in streams and filesystem functions are replacements
- Extension is old and not really kept up to date (bit rot)
- Extension is more low level

- stream_socket_server
- stream_socket_client

Processes are Black Magic

- Pipes
- STDIN, STDOUT, STDERR
- proc_open
- popen

Stream Contexts

- Parameters
- Options
- Modify or enhance a stream

- stream_context_set_param
- stream_context_set_option
- stream_context_create

Built In

Streams, Stream Transports, and Filters all available by default

Built in Streams

- file://
- http://
- ftp://
- data://
- glob://

Using Http

```php
<?php
// should be in format of $options[$stream_wrapper][$option]
$options = array('http' => array('method' => 'HEAD'));

// create our context
$context = stream_context_create();

file_get_contents('http://example.com', false, $context);

// our magically delicious headers
var_dump($http_response_headers);

// set our context as default, we can be evil after 5.3
stream_set_default_context($context);

include('http://example.com');
```

Extensions with Streams

- SSL
  - https://
  - ftps://
  - ssl://
  - tls://
- SSH
  - *ssh2.shell://*
  - *ssh2.exec://*
  - *ssh2.tunnel://*
  - *ssh2.sftp://*
  - *ssh2.scp://*
- Phar
  - phar://
- Zlib
  - compress.zlib://
  - zlib://
- Bzip
  - compress.bz2://

Using SSL (sockets)

```php
<?php
function safe_feof($fp, &$start = NULL) {
 $start = microtime(true);

 return feof($fp);
}

$start = NULL;
$timeout = ini_get('default_socket_timeout');

$fp = fsockopen('ssl://www.sandbox.paypal.com', 443, $errno,
                $errstr, 30);

//send request
fputs ($fp, 'Headers and Request Data');
while(!safe_feof($fp, $start) && (microtime(true) - $start) < $timeout)
    //get response
    $res = fgets ($fp, 1024);
    if ('VERIFIED' === $res) {
        // we did it, log it and send an email
    } elseif ('INVALID' === $res) {
        // something broke, log it and send an email
    }
}
fclose ($fp);
```

Using ssh2 streams

```php
<?php
$connection = ssh2_connect('example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

// clean out our remote directory
$stream = ssh2_exec($connection, 'rm -Rf /home/myfiles/*', false);
stream_set_blocking( $stream, true );
fclose($stream);

// stick new stuff in our remote directory
$sftp = ssh2_sftp($connection);
foreach ($files as $remote => $local) {
        $path = dirname($remote);
        if (!is_dir("ssh2.sftp://{$sftp}$path")) {
                mkdir("ssh2.sftp://{$sftp}$path", 0755, true);
        }
        if (file_exists($local)) {
                copy($local, "ssh2.sftp://{$sftp}$remote");
        }
}
```

Phar Streams

```php
<?php
$context = stream_context_create(array('phar' =>
                                    array('compress' => Phar::GZ)),
                                    array('metadata' =>
                                        array('user' => 'cellog')));
file_put_contents('phar://my.phar/somefile.php', 0, $context);

include 'phar://coollibrary.phar/internal/file.php';
header('Content-type: image/jpeg');

echo file_get_contents('phar:///coollibrary.phar/images/wow.jpg');
```

Built in Filters

- string filters
  - string.rot13
  - string.toupper
  - string.tolower
  - string.strip_tags

```php
<?php

$fp = fopen('php://output', 'w');
stream_filter_append($fp, 'string.rot13');
fwrite($fp, "This is a test.\n");

file_put_contents("php://filter/write=string.rot13/resource=example.txt",
                  "Hello World");

```

… Some more built in filters

- convert filters
  - convert.*
    - base64-encode
    - base64-decode
    - quoted-printable-encode
    - quoted-printable-decode
- dechunk
  - decode remote HTTP chunked encoding streams
- consumed
  - eats data (that's all it does)

Extension Filters

- bzip.compress and bzip.compress
- convert.iconv.*
- zlib.inflate and zlib.deflate
- mcrypt.* and mdecrypt.*

Built in Socket Transports

- tcp
- udp
- unix
- udg
- SSL extension
  - ssl
  - sslv2
  - sslv3
  - tls

PHP's Magic Special Voodoo

- *php://stdin*
- *php://stdout*
- *php://stderr*
- *php://output*
- *php://input*
- *php://filter* (5.0.0)
- *php://memory* (5.1.0)
- *php://temp* (5.1.0)

Why PHP streams rock

```php
<?php
$tempfile = tempnam(sys_get_temp_dir());

// get the file from FTP to local disk
$fh = ftp_connect('ftphost.com', 21);
ftp_login($fh, 'username', 'password');
ftp_get($fh,$tempfile, '/path/to/file.dat.gz');
ftp_close($fh);

// read data from local .gz into var
$gh = gzopen($tempfile, 'r');
$data = gzread($gh, 1000000);
gzclose($gh);
unlink($tempfile);

// write data to local .dat
file_put_contents('/local/copy/of/file.dat', $data);
```

```php
<?php
copy('compress.zlib://ftp://username:password@ftphost.com:21/path/to/file.dat.gz',
    '/local/copy/of/file.dat');
```

Custom Stuff

Userland Filters and Streams

Writing Custom Streams

- There are no interfaces
- Implement as though there were an interface
- Seekable is optional
- Flushable is optional
- Directory support is optional

Opening the Stream

Information

- fopen
- file_get_contents

- Return true or false
- $this->context will have any context metadata

Code

```
7     public function stream_open($path, $mode, $options, &$opened_path) {
8         echo $path;
9         // I should parse my path and check my $mode
10        // I should check my options
11        // $options & STREAM_USE_PATH means I fill in opened_path
12
13        return true;
14    }
15
```

Reading the Stream

Information

- fread
- fgets
- file_get_contents
- etc…

- Return string data or false
- $this->context will have any context metadata

Code

```
24      public function stream_read($count) {
25          $ret = 'My string';
26          $this->position += strlen($ret);
27          if ($this->position > 6) {
28                  return '';
29          }
30          return $ret;
31      }
```

Writing to the Stream

Information

- fwrite
- file_put_contents

- get in a string of data to deal with
- return how many bytes you wrote

Code

```
41      public function stream_write($data) {
42          $this->position += strlen($data);
43          return strlen($data);
44      }
```

The end of Data

Information

- feof
- file_get_contents
- fread
- etc…

- Return true or false
- $this->context will have any context metadata

Code

```
16    public function stream_eof() {
17        // You can access any class properties you placed during
18        // __construct and stream_open, but other calls are not
19        // guaranteed to have been made
20
21        return true;
22    }
```

Cleaning Up

Information

- fclose
- file_get_contents

- Don't return anything
- any cleanup should go here

Code

```
36
37      public function stream_close() {
38          echo "closing";
39      }
40
```

Stat

- fstat calls stream_stat
- EVERYTHING ELSE uses url_stat
- Good idea to do both

- Return an array of data identical to stat()

Seeking

- stream_seek
- stream_tell

Flush

- stream_flush

Directory Functionality

- mkdir
- rmdir
- dir_closedir
- dir_opendir
- dir_readdir
- dir_rewinddir

Extra stuff

- stream_lock
- stream_cast
- rename
- unlink

Writing Custom Filters

- Extend an internal class php_user_filter
- It's not abstract…
- Yes that's a horrible name
- Remember this pre-dates php 5.0 decisions

Setup

Information

- onCreate
- basically a constructor
- Called every time PHP needs a new filter (on every stream)

- return true or false

Code

```
7     /* Called when the filter is initialized */
8     function onCreate() {
9         $this->data = '';
10        return true;
11    }
```

Properties

- php_user_filter
  - $this->filtername
  - $this->params
  - $this->stream

Cleanup

Information

- onClose
- basically a destructor

- no return

Code

```
13      /* Called when the filter is deinitialized */
14      function onClose() {
15          $this->data = '';
16          return true;
17      }
18
```

Filter

Information

- MUST return
  - PSFS_PASS_ON
  - PSFS_FEED_ME
  - PSFS_ERR_FATAL
- You get buckets of data and do stuff to them

Code

```
2  function filter($in, $out, &$consumed, $closing)
3  {
4      while ($bucket = stream_bucket_make_writeable($in)) {
5          $bucket->data = strtolower($bucket->data);
6
7          $consumed += $bucket->datalen;
8          stream_bucket_append($out, $bucket);
9      }
10     return PSFS_PASS_ON;
11 }
```

Bucket Brigades

Filters and Buckets

- $in and $out are “bucket brigades” containing opaque “buckets” of data
- You can only touch buckets and brigades with the stream_bucket_* functions
- You get a bucket using stream_bucket_make_writeable

Real Uses

Use Case land – when streams make sense

The Requirements

- Data in s3
- Data locally during development
- Easy switch out if alternative storage is ever desired
- Storing image files

The Solution

- Existing Zend Framework Code
- Register the s3:// wrapper
- Use a configuration setting for the stream to use for all images on the system

The Requirements

- Store and edit template files in a database
- Have the snappiness of including from disk
- Minimal Configuration

The Solution

- db:// stream
- simple stream wrapper that looks for the template in the db, and writes it to the filesystem before returning the data
- The cached location is FIRST in the include path, so if it fails, the db stream gets hit

Requirements

- Talk to mercurial (hg binary)
- hg communicates via command line
- continue to pipe additional commands

The Solution

- Use proc_open to keep a pipe to the binary going
- Pass commands through stdin pipe as necessary
- Abstract this out to other binaries that are used by the system

Thanks!

- Elizabeth Marie Smith auroraeosrose@gmail.com
- http://php.net/streams
- http://php.net/filesystem
- http://ciaranmcnulty.com/blog/2009/04/simplifying-file-operations-using-php-stream-wrappers