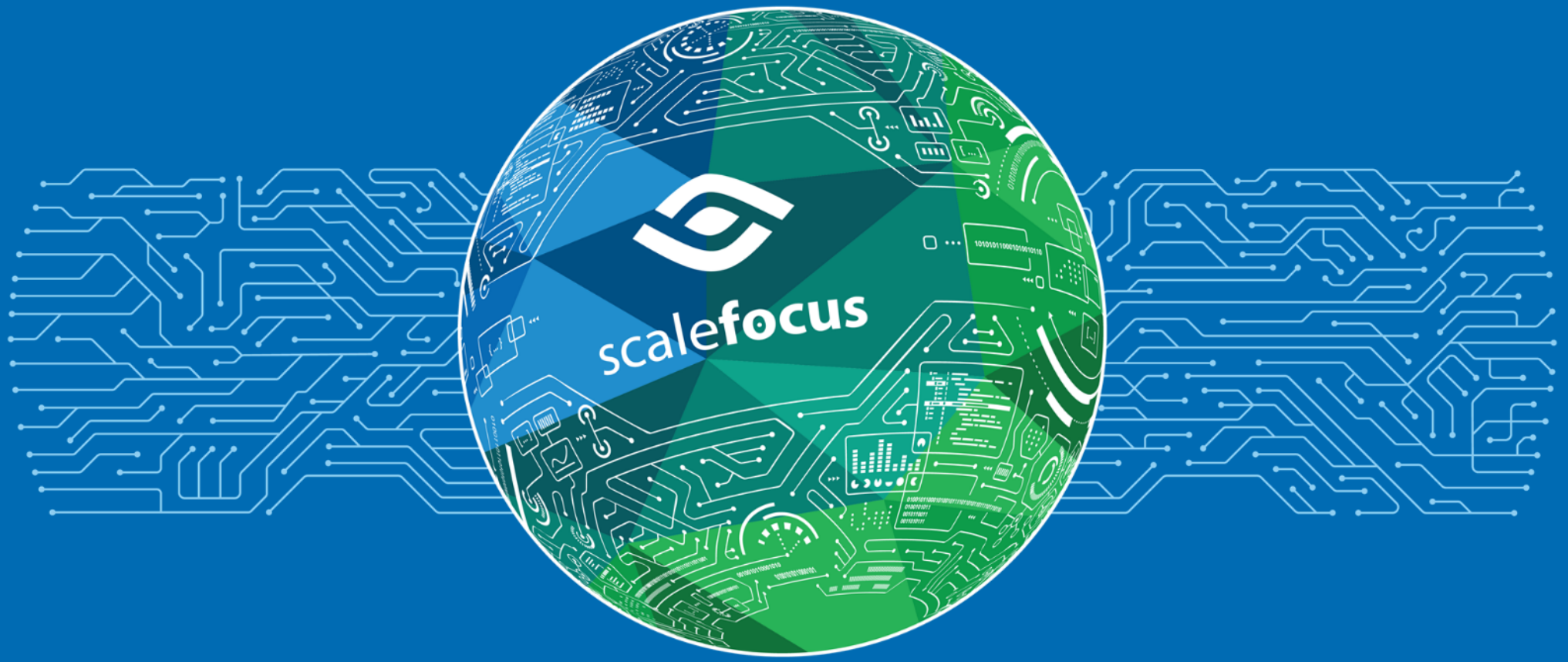

PHP 7.2 New Features

Presented by Georgi Georgiev

New Object Type

```php
<?php

function test(object $obj) : object
{
    return new SplQueue();
}

test(new StdClass());
```

Extension loading by name

Shared extensions no longer require their file extension (*.so* for Unix or *.dll* for Windows) to be specified

```
load_lib('openssl');
```

Abstract Method Overriding

```php
<?php

abstract class A
{
    abstract function test(string $s);
}
abstract class B extends A
{
    // overridden - still maintaining contravariance for parameters a
nd covariance for return
    abstract function test($s) : int;
}
```

Sodium is now a core extension

The modern Sodium cryptography library has now become a core extension in PHP.

Installation:

apt-get install libsodium-dev -> For Debian 8+, Ubuntu 15.04+
For windows:
https://windows.php.net/downloads/pecl/releases/libsodium/1.0.2/

Sodium is now a core extension Example

Lets give John a set of keys:

```
$johns_key_pair = sodium_crypto_box_keypair();
$johns_public_key = sodium_crypto_box_publickey($johns_key_pair);
$johns_private_key = sodium_crypto_box_secretkey($johns_key_pair);
```

Cool, let's do the same with Sasha

```
$sashas_key_pair = sodium_crypto_box_keypair();
$sashas_public_key = sodium_crypto_box_publickey($sashas_key_pair);
$sashas_private_key = sodium_crypto_box_secretkey($sashas_key_pair);
```

Sodium is now a core extension Example

Awesome, so John wants to send a message to Sasha, containing his most valued possession: Intel i7-7700HQ CPU. We can write this out as follows:

```
$priced_possession = 'Intel i7-7700HQ CPU';

$throw_off_bytes = random_bytes(SODIUM_CRYPTO_BOX_NONCEBYTES);

$encryption_key =
sodium_crypto_box_keypair_from_secretkey_and_publickey($johns_private_key,
$sashas_public_key);
$encrypted = sodium_crypto_box($priced_possession, $throw_off_bytes, $encryption_key);
```

If we run the following command: echo base64_encode($encrypted); we can see this:

EBQEZAsKp4pWULJApRrVIQvmImOZ6alFY3+T/UoESHKtKxM=

Password hashing with Argon2

```
password_hash('password', PASSWORD_ARGON2I, ['memory_cost' => 1<<17, 'time_cost'
=> 4, 'threads' => 2]);
```

PASSWORD_ARGON2_DEFAULT_MEMORY_COST = 1024 kiB
PASSWORD_ARGON2_DEFAULT_TIME_COST = 2
PASSWORD_ARGON2_DEFAULT_THREADS = 2

Support for extended operations in LDAP

Support for EXOP has been added to the LDAP extension.

LDAP:

```
$handle = ldap_connect('ldap://active.directory.server/');
$bind = ldap_bind($handle, 'user', 'expiredpass');

if ($bind) {
    if (ldap_get_option($handle, LDAP_OPT_DIAGNOSTIC_MESSAGE,
$extended_error)) {
        echo "Error Binding to LDAP: $extended_error";
    } else {
        echo "Error Binding to LDAP: No additional information is
available.";
    }
}
```

Support for extended operations in LDAP

ldap_exop:

ldap_exop (resource $link , string $reqoid [, string $reqdata = NULL [, array $serverctrls = NULL [, string &$retdata [, string &$retoid]]]]) : mixed

Support for extended operations in LDAP Example

```php
<?php
$ds = ldap_connect("localhost");  // assuming the LDAP server is on this host

if ($ds) {
    // bind with appropriate dn to give update access
    $bind = ldap_bind($ds, "cn=root, o=My Company, c=US", "secret");
    if (!$bind) {
      echo "Unable to bind to LDAP server";
      exit;
    }

    // Call WHOAMI EXOP
    $r = ldap_exop($ds, LDAP_EXOP_WHO_AM_I);

    // Parse the result object
    ldap_parse_exop($ds, $r, $retdata);
    // Output: string(31) "dn:cn=root, o=My Company, c=US"
    var_dump($retdata);

    // Same thing using $retdata parameter
    $success = ldap_exop($ds, LDAP_EXOP_WHO_AM_I, NULL, NULL, $retdata,
$retoid);
    if ($success) {
      var_dump($retdata);
    }

    ldap_close($ds);
} else {
    echo "Unable to connect to LDAP server";
}
?>
```

Address Information additions to the Sockets extension

socket_addrinfo_lookup()
socket_addrinfo_connect()
socket_addrinfo_bind()
socket_addrinfo_explain()

Address Information additions to the Sockets extension

```php
<?php
$addrinfo = socket_addrinfo_lookup('localhost', 2000, array('ai_family' => AF_INET, 'ai_socktype' => SOCK_STREAM));
$sockaddr = reset($addrinfo);
if (!$sockaddr) die ("No Valid Socket Types");
$sock = socket_addrinfo_bind($sockaddr);
// ^^ $sock is a socket resource that is bound to 127.0.0.1:2000 using TCP/IP ready for reading

var_dump(socket_addrinfo_explain($sockaddr));
/* Outputs:
array(5) {
  ["ai_flags"]=>
  int(0)
  ["ai_family"]=>
  int(2)
  ["ai_socktype"]=>
  int(1)
  ["ai_protocol"]=>
  int(6)
  ["ai_addr"]=>
  array(2) {
    ["sin_port"]=>
    int(2000)
    ["sin_addr"]=>
    string(9) "127.0.0.1"
  }
}
*/
```

Address Information additions to the Sockets extension

```
socket_addrinfo_connect ( resource $addr ) : resource
```

Address Information additions to the Sockets extension

```
socket_addrinfo_bind ( resource $addr ) : resource
```

Address Information additions to the Sockets extension

```
socket_addrinfo_explain ( resource $addr ) : array
```

Parameter type widening

```
interface A
{
    public function Test(array $input);
}

class B implements A
{
    public function Test($input){} // type omitted for $input
}
```

Allow a trailing comma for grouped namespaces

```php
<?php

use Foo\Bar\{
    Foo,
    Bar,
    Baz,
};
```

proc_nice() support on Windows

proc_nice (int $increment) : bool

```
<?php

//decrease niceness
proc_nice(19);

//kill child process to "reset" niceness
posix_kill( getmypid(), 28 );

?>
```

Enhancements to the EXIF extension

Samsung
DJI
Panasonic
Sony
Pentax
Minolta
Sigma/Foveon
AGFA
Kyocera
Ricoh
Epson

Enhancements to the EXIF extension
Example exif_read_data()

```php
<?php
// Open a the file, this should be in binary mode
$fp = fopen('/path/to/image.jpg', 'rb');

if (!$fp) {
    echo 'Error: Unable to open image for reading';
    exit;
}

// Attempt to read the exif headers
$headers = exif_read_data($fp);

if (!$headers) {
    echo 'Error: Unable to read exif headers';
    exit;
}

// Print the 'COMPUTED' headers
echo 'EXIF Headers:' . PHP_EOL;

foreach ($headers['COMPUTED'] as $header => $value) {
    printf(' %s => %s%s', $header, $value, PHP_EOL);
}
?>
```

```
EXIF Headers:
 Height => 576
 Width => 1024
 IsColor => 1
 ByteOrderMotorola => 0
 ApertureFNumber => f/5.6
 UserComment =>
 UserCommentEncoding => UNDEFINED
 Copyright => Denis
 Thumbnail.FileType => 2
 Thumbnail.MimeType => image/jpeg
```

SQLite3 allows writing BLOBs

public SQLite3::openBlob (string $table , string $column , int $rowid [, string $dbname = "main" [, int $flags = SQLITE3_OPEN_READONLY]]) : resource

SQLite3 allows writing BLOBs Example

```php
<?php
$conn = new SQLite3(':memory:');
$conn->exec('CREATE TABLE test (text text)');
$conn->exec("INSERT INTO test VALUES (zeroblob(36))");
$stream = $conn-
>openBlob('test', 'text', 1, 'main', SQLITE3_OPEN_READWRITE);
for ($i = 0; $i < 3; $i++) {
    fwrite($stream,  "Lorem ipsum\n");
}
fclose($stream);
echo $conn->querySingle("SELECT text FROM test");
$conn->close();
?>
```

Enhancements to the ZIP extension

The ZipArchive class now implements the Countable interface.
//ZipArchive::count

Additional explanations

1. can a normal class, override abstract methods, or only abstract classes can do it - Only abstract methods can, tested it with the same example removed the abstract method from Class B and it threw an error: **Fatal error**: Declaration of B::test($s): int must be compatible with A::test(string $s) in **C:\xampp\htdocs\test\test.php** on line **13**

2. sodium crypt alternatives in previous versions - OpenSSL, Crack, CSPRNG , Mcrypt, the others in the link are for hashing
https://www.php.net/manual/en/refs.crypto.php

3. in argon is the time cost in seconds or ms - Time Cost represents the number of times the the hash algorithm will be run

4. check if by omitting the type, the method also accepts different types or if you can change the type at all – After omitting, the method will accept different types of arguments.
https://blog.kelunik.com/2018/01/23/parameter-type-widening.html

5. explain if the connection to the blob is via socket – When we use SQLite3::openBlob we are opening a stream resource to read or write a BLOB. Streams were introduced with PHP 4.3.0 as a way of generalizing file, network, data compression, and other operations which share a common set of functions and uses. In its simplest definition, a stream is a resource object which exhibits streamable behavior. That is, it can be read from or written to in a linear fashion, and may be able to fseek() to an arbitrary locations within the stream.
https://www.sitepoint.com/%EF%BB%BFunderstanding-streams-in-php/

Thank You