

Talking About PSR-7

전창완 (im@wani.kr)

Contents

1. PSR-7은 왜 만들어졌을까?
2. PSR-7은 어떻게 생겼나?
3. PSR-7은 앞으로 어떻게 될까?

1

PSR-7은 왜 만들어졌을까?



mordern
php user group

PHP..?

PHP(PHP: Hypertext Preprocessor)는 프로그래밍 언어의 일종이다. 원래는 동적 웹 페이지를 만들기 위해 설계되었으며 이를 구현하기 위해 PHP로 작성된 코드를 HTML 소스 문서 안에 넣으면 PHP 처리 기능이 있는 웹 서버에서 해당 코드를 인식하여 작성자가 원하는 웹 페이지를 생성한다. (중략)

〈출처 : ko.wikipedia.org/wiki/PHP〉

다른 언어와 실행시점이 다르다.

Modern0이고 뭐고..
다 빼고 한번 살펴봅시다.

```
var http = require('http');  
  
var server = http.createServer(function (request, response) {  
    response.end("hello world");  
});  
  
server.listen(8000);
```

Node.js : 모든 연결을 다 제어

```
<?php  
echo "hello world";
```

PHP : 단 하나의 연결만을 제어

다양한 전역변수..

\$_GET, \$_POST, \$_SERVER, \$_FILES,
\$_COOKIE, \$_SESSION...



mordern
php user group

그뿐이라!

```
apache_request_headers();  
http_get_request_body();  
fopen('php://input');
```

OOP가 등장하고..
프레임워크 탄생!!!



mordern
php user group

기존의 PHP로 들어오는 데이터가
너무 파편화 되어있어!!



Request ¶

The most common way to create a request is to base it on the current PHP global variables with `createFromGlobals()`:

```
1 use Symfony\Component\HttpFoundation\Request;
2
3 $request = Request::createFromGlobals();
```

which is almost equivalent to the more verbose, but also more flexible, `__construct()` call:

```
1 $request = new Request(
2     $_GET,
3     $_POST,
4     array(),
```



phalcon

Class **Phalcon\Http\Request**

implements *Phalcon\Http\RequestInterface*, *Phalcon\Di\InjectionAwareInterface*

Encapsulates request information for easy and secure access from application controllers. The request object is a simple value object that is passed between the dispatcher and controller classes. It packages the HTTP request environment.

```
<?php

$request = new \Phalcon\Http\Request();
if ($request->isPost() == true) {
    if ($request->isAjax() == true) {
        echo 'Request was made using POST and AJAX';
    }
}
```



phalcon



대부분의 프레임워크가
Request/Response방식을 사용 중.

What if we could share code between
these applications and frameworks at
the HTTP level?

– Igor, Beau, and Christoph

2012년 12월 31일

HTTP Message
Chris Wilkinson

2014년 8월 29일

PSR-7 HTTP Message – Retracted
Michael Dowling

2012년 3월 24일

HTTP Client interface
Benjamin Eberlei

2014년 1월 28일

PSR-7 HTTP Message – Draft
Michael Dowling

2014년 9월 26일

PSR-7 HTTP Message Interfaces
Matthew Weier O’Phinney



[VOTE][Accept] PSR-7: HTTP message interfaces

작성자 27명의 게시물 31개  

★ Beau Simensen	The required review period has passed. During the review period a number of issues were raised	3월 20일
★ Josh Lockhart	+1 Slim	3월 20일
★ Beau Simensen	+1 Sculpin	3월 20일
★ CalEvans	+1 from Community =C=	3월 20일
★ Larry Garfield	+1 --Larry Garfield, Drupal	3월 20일
★ Evert Pot	-1 from sabre/dav	3월 20일
★ Taylor Otwell	+1 Laravel	3월 20일
★ Michael Dowling	+1 AWS	3월 20일
★ Brett Bieber	+1 from PEAR	3월 20일
★ Pádraic Brady	+1 on behalf of Zend Framework Paddy	3월 20일
★ Matteo Beccati	+1 from Revive Adserver	3월 20일
★ Philip Sturgeon	I'm just gonna put this right here. *PSR-7: * https://github.com/php-fig/fig-	3월 21일
★ Emil Rosendahl	That sure is a cute kitten!	3월 21일
★ Andres Gutierrez	+1 from Phalcon	3월 21일
★ Claudio Ludovico Panetta	+1	3월 21일
★ Beau Simensen	As a reminder, this thread is for casting votes by voting representatives of member projects only. If	3월 21일
★ Lukas Kahwe Smith	+1 from Jackalope regards, Lukas	3월 21일
★ Kris Wallsmith	+1 from Assetic!	3월 21일
★ Jordi Boggiano	+1 Composer	3월 22일

2015년 5월 18일

PSR-7 Accept!



mordern
php user group

2

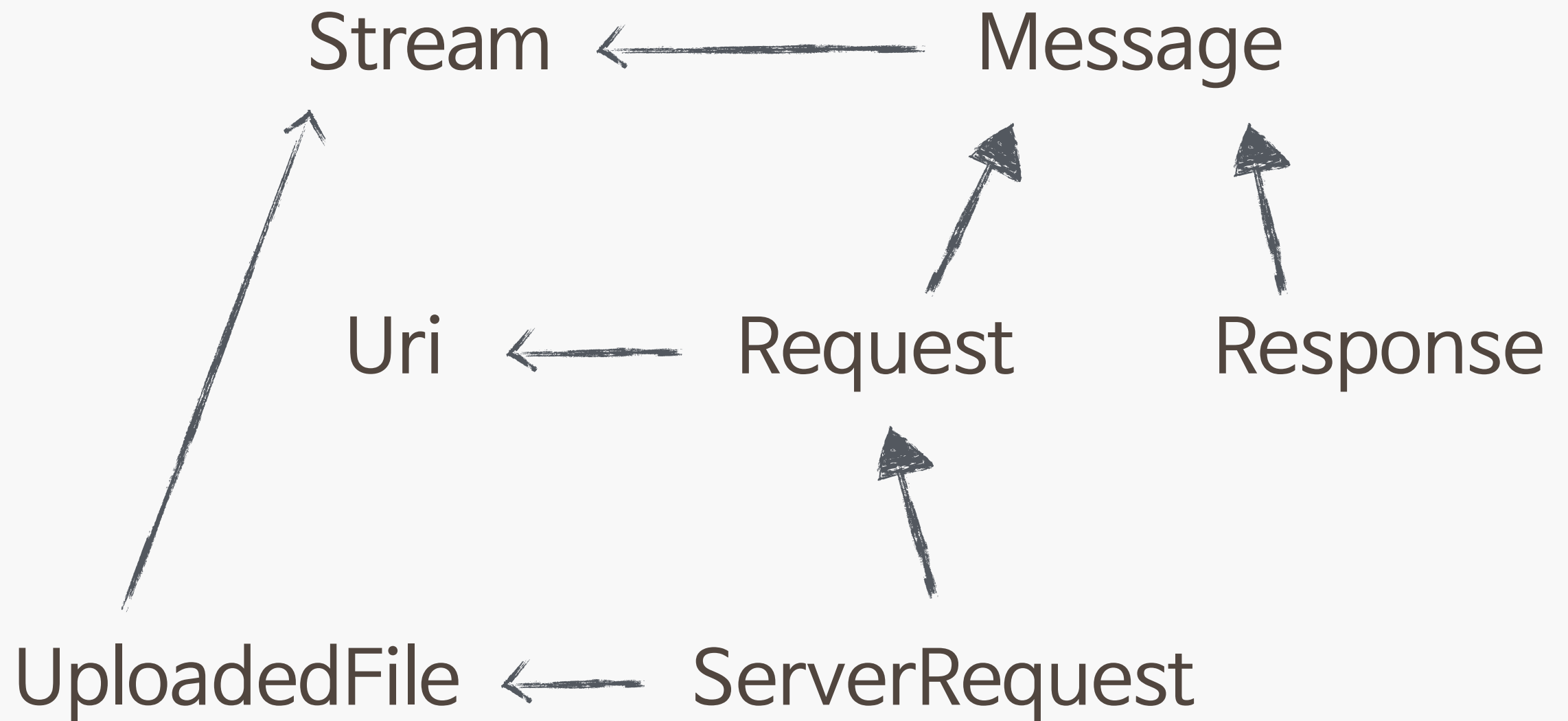
PSR-7은 어떻게 생겼나?



mordern
php user group

7개의 인터페이스

- MessageInterface
- RequestInterface
- ResponseInterface
- ServerRequestInterface
- StreamInterface
- UploadedFileInterface
- UriInterface



공통사항

- get~ 과 with~으로 구성.
- with~ 는 setter가 아님, 복제된 객체를 반환. (주의)

StreamInterface

- `__toString`
- `close` / `detach` / `getSize`
- `tell` / `eof` / `rewind` / `isSeekable` / `seek` : 탐색 관련
- `isWritable` / `write` : 쓰기 관련
- `isReadable` / `read` / `getContents` : 읽기 관련
- `getMetadata` : 메타값 관련

MessageInterface

- `getProtocolVersion / withProtocolVersion` : HTTP 버전 관리
- `getHeaders / hasHeader / getHeader /
getHeaderLine / withHeader / withAddedHeader /
withoutHeader` : 헤더 관리
- `getBody / withBody` : 메시지 관리



modern
php user group

UriInterface

- **getScheme / withScheme** : Scheme (ex. HTTP, HTTPS)
- **getAuthority** : Authority (ex. user:password@wani.kr:8080)
- **getUserInfo / withUserInfo** : UserInfo (ex. user:password):
- **getHost / withHost** : Host
- **getPort / withPort** : 표준이 아닌 포트를 반환, 80, 443이 아닌 경우에만.
- **getPath / withPath** : Path (ex. /abc/def)
- **getQuery / withQuery** : Query (ex. abc=def&hello=world)
- **getFragment / withFragment** : Fragment (ex. #hello-world)

RequestInterface

- **getRequestTarget / withRequestTarget** : RequestTarget관리 (path + query string)
- **getUri / withUri** : URI관련
- **getMethod / withMethod** : 메서드관리

UploadedFileInterface

- `$_FILES`를 대입해서 보면 이해하기 좋음.
- **getStream** : `$_FILES['name']['tmp_name']`에 해당하는 파일 Stream
- **moveTo** : `move_uploaded_file`을 실행:
- **getSize** : `$_FILES['name']['size']`:
- **getError** : `$_FILES['name']['error']`:
- **getClientFilename** : `$_FILES['name']['name']`:
- **getClientMediaType** : `$_FILES['name']['type']`:



ServerRequestInterface

- **getServerParams** : `$_SERVER`
- **getCookieParams** / **withCookieParams** : `$_COOKIE`
- **getQueryParams** / **withQueryParams** : `$_GET`
- **getUploadedFiles** / **withUploadedFiles** : `$_FILES`, 중요한것은 여기서 반환형이 Tree구조로 반환함.

<https://github.com/Wandu/Http/blob/master/tests/Factory/UploadedFileFactoryTest.php>

- **getParsedBody** / **withParsedBody** : `$_POST`
- **getAttributes** / **getAttribute** / **withAttribute** / **withoutAttribute**
: 외부에서 Match에 대한 값을 주입하기 위한 매서드들.

ResponseInterface

- **getStatusCode** : 200, 301, 404, 500...
- **getReasonPhrase** : OK, MovedPermanently, Not Found, Internal Server Error...
- **withStatus** : 위 두가지를 한번에 적용.



mordern
php user group

실제로 사용하고 싶다면..

github.com/zendframework/zend-diactoros

github.com/phly/http

github.com/Wandu/Http




```
<?php
namespace Your\Project;

use Wandu\Http\Factory\ServerRequestFactory;
use Wandu\Http\Sender\ResponseSender;

$basePath = dirname(__DIR__);
require $basePath . '/vendor/autoload.php';

$app = new Application($basePath);
$response = $app->execute(ServerRequestFactory::fromGlobals());
ResponseSender::send($response);
```

```
<?php
namespace Your\Project\Controller;

use Closure;
use Psr\Http\Message\ServerRequestInterface;

class Administrator extends BaseController
{
    public function auth(ServerRequestInterface $request, Closure $next)
    {
        return $next($request);
    }

    public function index()
    {
        $this->session->set('hello', 'world!!!!');
        return Response::plain($this->view->render('admin/index'));
    }
}
```

그렇지만 예제는.. Wandu/Http


3 PSR-7은 앞으로 어떻게 될까?

[VOTE][Accept] PSR-7: HTTP message interfaces

작성자 27명의 게시물 31개  

★ Beau Simensen	The required review period has passed. During the review period a number of issues were raised	3월 20일
★ Josh Lockhart	+1 Slim	3월 20일
★ Beau Simensen	+1 Sculpin	3월 20일
★ CalEvans	+1 from Community =C=	3월 20일
★ Larry Garfield	+1 --Larry Garfield, Drupal	3월 20일
★ Evert Pot	-1 from sabre/dav	3월 20일
★ Taylor Otwell	+1 Laravel	3월 20일
★ Michael Dowling	+1 AWS	3월 20일
★ Brett Bieber	+1 from PEAR	3월 20일
★ Pádraic Brady	+1 on behalf of Zend Framework Paddy	3월 20일
★ Matteo Beccati	+1 from Revive Adserver	3월 20일
★ Philip Sturgeon	I'm just gonna put this right here. *PSR-7: * https://github.com/php-fig/fig-	3월 21일
★ Emil Rosendahl	That sure is a cute kitten!	3월 21일
★ Andres Gutierrez	+1 from Phalcon	3월 21일
★ Claudio Ludovico Panetta	+1	3월 21일
★ Beau Simensen	As a reminder, this thread is for casting votes by voting representatives of member projects only. If	3월 21일
★ Lukas Kahwe Smith	+1 from Jackalope regards, Lukas	3월 21일
★ Kris Wallsmith	+1 from Assetic!	3월 21일
★ Jordi Boggiano	+1 Composer	3월 22일

Laravel (dev)

**Laravel**
@laravelphp

팔로우

Thanks to [@symfony](#)'s work in bringing PSR-7 to HTTP Foundation, Laravel and Lumen will also have PSR-7 support in 5.1!


번역 보기

Little Rock, AR

← ↻ ★ ...

106
리트윗

105
관심글



오전 11:57 - 2015년 5월 30일

2015년 6월 9일 공개 예정

Slim 3.0 (planning)

Version 3.0

The next framework release will be version 3.0. This is a major update that touches all parts of the framework. Here are a few highlights:

Dependency injection

The `\Slim\App` class will extend Pimple so we can easily inject third-party components into a Slim application or override many of Slim's internal objects such as the request, response, or view objects.

PSR-7 support

~~Slim's HTTP request and response abstractions will support PSR-7.~~ This means their interfaces will differ significantly from previous releases. In the past, each Slim application had one request object and one response object that were passed by reference throughout the entire application.

Version 3, however, treats the request and response objects as *value objects*. Each middleware layer and application route will receive the most current request and response objects as arguments. Each middleware layer and route callback is responsible for *returning* an updated HTTP response object.

The HTTP request and response objects are *immutable*, too. You must use the appropriate `withStatus()`, `withHeader()`, `withBody()`, etc. request and response object methods to create and return a new request or response object with the specified changes. You can read more about the new interface in the PSR-7 documentation at <https://github.com/php-fig/fig-standards/blob/master/proposed/http-message.md>.

Zend Framework 3 (dev)

Announcing the Zend Framework 3 Roadmap

The most often-asked questions we get around the Zend Framework project include: Where is Zend Framework heading? When will Zend Framework 3 be released? What changes and enhancements should we expect?

Since inception, our goal for Zend Framework has been to further the art of PHP and ensure our users concentrate on the business logic of their application rather than wasting time reinventing the plumbing. The plumbing is Zend Framework's job. We have continued to evolve ZF with best-in-class web development practices, and have innovated in areas where we saw gaps; as an example, we observed developers struggling with API development, which led us to create the Apigility project on top of ZF2.

We have built an incredibly powerful framework with Zend Framework 2 that met its key goals of flexibility, consistency and testability. However, the world has changed since ZF2 was released, and the project needs to move with the times. With that in mind, we have gathered feedback from our users and core contributors to map the path forward.

Zend Framework 3 will be an evolution from ZF2, concentrating on simplicity, reusability, and performance.

What's involved?

- **Separating components into individual, versioned projects.** This enables broader re-use and higher velocity of innovation.
- Strong emphasis on HTTP messages, with Matthew leading the [PSR-7 specification](#).
- ~~Updating our existing full-stack MVC framework to depend on the newly independent components for better reuse and simplicity. ZF2 MVC applications will have a documented upgrade path to ZF3 requiring~~

Symfony (support)

PSR-7 Support in Symfony is Here

May 30, 2015



[Ryan Weaver](#)

Less than 2 weeks ago, the PHP community roundly accepted [PSR-7](#), giving PHP a common set of *HTTP Message Interfaces*. This has *huge* potential for interoperability and standardization across all of PHP. This is especially true for [middleware](#): functions that hook into the request-response process. In the future, a middleware written around these new interfaces could be used in *any* framework.

Introducing the PSR HTTP Message Bridge ¶

Today, a *huge* number of projects use Symfony's [Request](#) and [Response](#) classes (via the [HttpFoundation](#) component), including Laravel, Drupal 8 and StackPHP. This has given us a base for HTTP Message standardization over the last 4 years, before the community was ready to discuss an official interface. Promoting interoperability is at the core of Symfony's philosophy.

Symfony (support)

대충 요약하자면, Symfony HttpFoundation이 사실상 표준화가 되어버린 상황이라 PSR-7 Bridge를 지원하겠다.

```
use Symfony\Bridge\PsrHttpMessage\Factory\HttpFoundationFactory;

$httpFoundationFactory = new HttpFoundationFactory();

// convert a Request
// $psrRequest is an instance of Psr\Http\Message\ServerRequestInterface
$symfonyRequest = $httpFoundationFactory->createRequest($psrRequest);

// convert a Response
// $psrResponse is an instance of Psr\Http\Message\ResponseInterface
$symfonyResponse = $httpFoundationFactory->createResponse($psrResponse);
```



Guzzle (include)

Guzzle and PSR-7

Guzzle utilizes PSR-7 as the HTTP message interface. This allows Guzzle to work with any other library that utilizes PSR-7 message interfaces.

Guzzle is an HTTP client that sends HTTP requests to a server and receives HTTP responses. Both requests and responses are referred to as messages.

Guzzle relies on the `guzzlehttp/psr7` Composer package for its message implementation of PSR-7.

You can create a request using the `GuzzleHttp\Psr7\Request` class:

```
use GuzzleHttp\Psr7\Request;

$request = new Request('GET', 'http://httpbin.org/get');

// You can provide other optional constructor arguments.
$headers = ['X-Foo' => 'Bar'];
$body = 'hello!';
$request = new Request('PUT', 'http://httpbin.org/put', $headers, $body);
```

You can create a response using the `GuzzleHttp\Psr7\Response` class:

그외에..

Drupal 8 (예정)
CakePHP (예정)
...

어쨌든 사건이지만..



mordern
php user group

한번 익혀두면 프레임워크를 바꿔서 사용하더라도
진입장벽이 크게 높아지지 않을 듯..

Phalcon의 동향은 안나오지만...

Phalcon도 투표에 참여했음.

PHP extension 형태로 나오게 되면 PHP 표준 라이브러리(spl)처럼 자리 잡게 될수도 있지 않을까?



modern
php user group

참고자료

<http://beau.io/talks/2015/05/16/hello-psr-7-phpday-2015>

<http://www.php-fig.org/psr/psr-7>

<http://ko.wikipedia.org/wiki/PHP>

..꽃



mordern
php user group