URI Documentation

Jon Cram

Aug 21, 2020

First Steps

The webginition/uri package models, normalizes, parses and compares URLs.

The *overview* provides a high-level introduction to the package classes.

You might want to skip straight to the *Uri usage guide* or the *Normalizer usage guide*.

CHAPTER 1

Overview

1.1 Url

A `Uri` models a URL, providing component access and modification through the PSR7 UriInterface.

```php
<?php

use webignition\Uri\Uri;

$uri = new Uri('http://example.com/path?query#fragment');

$uri->getScheme();
// "http"

$uri->getQuery();
// "query"

$modifiedUri = $uri
    ->withScheme('https')
    ->withPath('/modified-path')
    ->withQuery('foo=bar')
    ->withFragment('');
(string) $modifiedUri;
// https://example.com/modified-path?foo=bar
```

Read the *Uri usage* guide for more detail.

1.2 Normalizer

The `Normalizer` can apply any combination of sixteen normalizations to any `UriInterface` implementation.

```php
<?php

use webignition\Uri\Normalizer;

$uri = new Uri('http://example.com/path?c=cow&a=apple&b=bear#fragment');

$normalizedUri = Normalizer::normalize(
    $uri,
    Normalizer::SORT_QUERY_PARAMETERS | Normalizer::REMOVE_FRAGMENT
);

(string) $normalizedUri;
// "http://example.com/path?a=apple&b=bear&c=cow"
```

Flags:

- *capitalize percent encoding*
- *decode unreserved characters*
- *convert empty http path*
- *remove default file host*
- *remove port host*
- *remove path dot segments*
- *convert host unicode to punycode*
- *reduce duplicate path slashes*
- *sort query parameters*
- *add path trailing slash-*
- *remove user info*
- *remove fragment*
- *remove www sub-domain*

Options:

- *specify a default scheme*
- *remove filenames from path by pattern*
- *remove query parameters by pattern*

Read the *Normalizer usage* guide for more detail.

1.3 Parser

The `Parser` transforms a URL string into an array of component parts. Useful for direct access to raw URL components.

The parser is used internally by `Uri::create()` and isn't needed for normal URL usage.

```php
<?php

use webignition\Uri\Parser;
```

(continues on next page)

(continued from previous page)

```php
$components = Parser::parse('https://example.com:8080/path?query#fragment');

$components[Parser::COMPONENT_SCHEME];
// "https"

$components[Parser::COMPONENT_HOST];
// "example.com"
```

CHAPTER 2

Requirements and Installation

2.1 Requirements

- PHP 7.2.0 or greater
- composer

2.2 Installation

```
composer require webignition/uri ^1
```

CHAPTER 3

URI Model

A `Uri` models a URL, providing component access and modification through the PSR7 UriInterface.

Minimal non-optional RFC 3986 normalization is applied by default:

- converts scheme to lowercase
- converts host to lowercase
- removes the default port

3.1 Creating a URI

A new `Uri` instance is created by passing a URL string to the constructor:

```php
<?php

use webignition\Uri\Uri;

$uri = new Uri('https://example.com');
```

3.2 Component Access

```php
<?php

use webignition\Uri\Uri;

$uri = new Uri('https://user:password@example.com:8080/path?query#fragment');

$uri->getScheme();
// "https"
```

(continues on next page)

(continued from previous page)

```php
$uri->getUserInfo();
// "user:password"

$uri->getHost();
// "example.com"

$uri->getPort();
// 8080

$uri->getAuthority();
// "user:password@example.com:8080"

$uri->getPath();
// "/path"

$uri->getQuery();
// "query"

$uri->getFragment();
// "fragment"
```

3.3 Component Modification

The `Uri::with*()` are used to set components. A `Uri` is immutable. The return value is a new `Uri` instance.

```php
<?php

use webignition\Uri\Uri;

$uri = new Uri('https://user:password@example.com:8080/path?query#fragment');
(string) $uri;
// "https://user:password@example.com:8080/path?query#fragment"

$uri = $uri->withScheme('http');
(string) $modifiedUri;
// "http://user:password@example.com:8080/path?query#fragment"

$uri = $uri->withUserInfo('new-user', 'new-password');
(string) $modifiedUri;
// "http://new-user:new-password@example.com:8080/path?query#fragment"

$uri = $uri->withUserInfo('');
(string) $modifiedUri;
// "http://example.com:8080/path?query#fragment"

$uri = $uri->withHost('new.example.com');
(string) $modifiedUri;
// "http://new.example.com:8080/path?query#fragment"

$uri = $uri->withPort(null);
(string) $modifiedUri;
// "http://new.example.com/path?query#fragment"

$uri = $uri->withPath('');
```

(continues on next page)

(continued from previous page)

```
(string) $modifiedUri;
// "http://new.example.com?query#fragment"

$uri = $uri->withQuery('');
(string) $modifiedUri;
// "http://new.example.com#fragment"

$uri = $uri->withFragment('');
(string) $modifiedUri;
// "http://new.example.com"
```

3.4 Non-Optional Normalization

```
<?php

use webignition\Uri\Uri;

$uri = new Uri('HTTPS://EXAMPLE.com:443');

$uri->getScheme();
// "https"

$uri->getHost();
// "example.com"

$uri->getPort();
// null
```

CHAPTER 4

Normalizer

The `Normalizer` can apply any combination of sixteen normalizations to any `UriInterface` implementation.

Normalizations are specified through either *flags* or *options*. Flags are for normalizations that can be turned or turned off (you either want it or you don't). Options are for normalizations that act on one or more variables that you get to choose.

Flags:

- *capitalize percent encoding*
- *decode unreserved characters*
- *convert empty http path*
- *remove default file host*
- *remove port host*
- *remove path dot segments*
- *convert host unicode to punycode*
- *reduce duplicate path slashes*
- *sort query parameters*
- *add path trailing slash*-
- *remove user info*
- *remove fragment*
- *remove www sub-domain*

Options:

- *specify a default scheme*
- *remove filenames from path by pattern*
- *remove query parameters by pattern*

4.1 Capitalize Percent Encoding

Convert percent-encoded triplets (such as `%3A`) to uppercase. Letters within a percent-encoded triplet are case-insensitive.

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('http://example.com/path%2fvalue');
$normalizedUri = Normalizer::normalize($uri, Normalizer::CAPITALIZE_PERCENT_ENCODING);

(string) $normalizedUri;
// "http://example.com/path%2Fvalue"
```

4.2 Decode Unreserved Characters

Convert percent-encoded characters that have no special meaning to their unencoded equivalents.

Decodes encoded forms of: `ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-._~`

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('http://example.com/%75%72%6C');
$normalizedUri = Normalizer::normalize($uri, Normalizer::DECODE_UNRESERVED_
↪CHARACTERS);

(string) $normalizedUri;
// "http://example.com/url"
```

4.3 Convert Empty HTTP Path

Applies a path of `/` where the path is empty and the scheme is `http` or `https`.

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('http://example.com');
$normalizedUri = Normalizer::normalize($uri, Normalizer::CONVERT_EMPTY_HTTP_PATH);

(string) $normalizedUri;
// "http://example.com/"
```

4.4 Remove Default File Host

Removes the host of `localhost` from a `file://` url.

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('file://localhost/path');
$normalizedUri = Normalizer::normalize($uri, Normalizer::REMOVE_DEFAULT_FILE_HOST);

(string) $normalizedUri;
// "file:///path"
```

4.5 Remove Default Port

Removes the port if it matches the default port for the scheme.

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('http://example.com:80');
$normalizedUri = Normalizer::normalize($uri, Normalizer::REMOVE_DEFAULT_PORT);

(string) $normalizedUri;
// "http://example.com"

$uri = new Uri('https://example.com:443');
$normalizedUri = Normalizer::normalize($uri, Normalizer::REMOVE_DEFAULT_PORT);
(string) $normalizedUri;
// "https://example.com"
```

4.6 Remove Path Dot Segments

The `.` and `..` path segments have a special meaning. These segments are removed and the path is re-written to be equivalent.

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('http://example.com/a/b/c/./../../g');
$normalizedUri = Normalizer::normalize($uri, Normalizer::REMOVE_PATH_DOT_SEGMENTS);

(string) $normalizedUri;
// "http://example.com/a/g"
```

4.7 Convert Host Unicode to Punycode

Unicode hosts containing non-ascii characters are converted to the punycode equivalent.

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('http://.example.com/');
$normalizedUri = Normalizer::normalize($uri, Normalizer::CONVERT_HOST_UNICODE_TO_
↪PUNYCODE);

(string) $normalizedUri;
// "http://xn--g6h.example.com/"
```

4.8 Reduce Duplicate Path Slashes

Reduces occurrences of multiple slashes in the path to single slashes.

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('http://example.com///path//');
$normalizedUri = Normalizer::normalize($uri, Normalizer::REDUCE_DUPLICATE_PATH_
↪SLASHES);

(string) $normalizedUri;
// "http://example.com/path/"
```

4.9 Sort Query Parameters

Alphabetically sorts query parameters by key.

Sorting is neither locale- nor unicode-aware. The purpose is to be able to compare URLs in a reproducible way.

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('http://example.com?c=cow&a=apple&b=bear');
$normalizedUri = Normalizer::normalize($uri, Normalizer::SORT_QUERY_PARAMETERS);

(string) $normalizedUri;
// "http://example.com?a=apple&b=bear&c=cow"
```

4.10 Add Path Trailing Slash

Add a trailing slash to the path if not present.

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('http://example.com');
$normalizedUri = Normalizer::normalize($uri, Normalizer::ADD_PATH_TRAILING_SLASH);

(string) $normalizedUri;
// "http://example.com/"
```

4.11 Remove User Info

Remove user credentials.

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('http://user:password@example.com');
$normalizedUri = Normalizer::normalize($uri, Normalizer::REMOVE_USER_INFO);

(string) $normalizedUri;
// "http://example.com"
```

4.12 Remove Fragment

Remove fragment component.

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('http://example.com#fragment');
$normalizedUri = Normalizer::normalize($uri, Normalizer::REMOVE_FRAGMENT);

(string) $normalizedUri;
// "http://example.com"
```

4.13 Remove www Sub-domain

Remove the www sub-domain.

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('http://www.example.com');
$normalizedUri = Normalizer::normalize($uri, Normalizer::REMOVE_WWW);

(string) $normalizedUri;
// "http://example.com"
```

4.14 Specify a Default Scheme

Specify a default scheme to be applied if none is present.

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('//www.example.com');
$normalizedUri = Normalizer::normalize($uri, Normalizer::NONE, [
    Normalizer::OPTION_DEFAULT_SCHEME => 'http',
]);

(string) $normalizedUri;
// "http://example.com"
```

4.15 Remove Filenames From Path By Pattern

Remove the filename from the path component. Removal is defined through one or more patterns.

Useful for stripping common default filenames such as `index.html`, `index.js` or `default.asp`.

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('http//www.example.com/index.html');
$normalizedUri = Normalizer::normalize($uri, Normalizer::NONE, [
    Normalizer::OPTION_REMOVE_PATH_FILES_PATTERNS => Normalizer::REMOVE_INDEX_FILE_
↪PATTERN,
]);

(string) $normalizedUri;
// "http://example.com/"
```

4.16 Remove Query Parameters By Pattern

Remove query parameters where the parameter key matches one of a set of patterns.

Useful for stripping query parameters considered by you to be irrelevant to the canonical form of a URL.

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('http//www.example.com?x=1&y=2&utm_source=facebook&utm_medium=18');
$normalizedUri = Normalizer::normalize($uri, Normalizer::NONE, [
    Normalizer::OPTION_REMOVE_QUERY_PARAMETERS_PATTERNS => [
        '/^utm_\w+/i',
    ],
]);

(string) $normalizedUri;
// "http://example.com?page=1&category=2"
```

CHAPTER 5

Applying Multiple Normalizations

You can apply any number of the sixteen normalizations when normalizing a `UriInterface` instance.

5.1 Multiple Flags

Combine flags using the bitwise `|` operator.

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('http://example.com?b=bear&a=apple#fragment');
$normalizedUri = Normalizer::normalize(
    $uri,
    Normalizer::SORT_QUERY_PARAMETERS | Normalizer::REMOVE_FRAGMENT
);

(string) $normalizedUri;
// "http://example.com?a=apple&b=bear"
```

5.2 Options With No Other Normalizations

To apply one or more options but no other normalizations, call `Normalizer::normalize()` with `Normalizer::NONE` as the `flags` argument.

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;
```

(continues on next page)

(continued from previous page)

```php
$uri = new Uri('http//www.example.com?x=1&y=2&utm_source=facebook&utm_medium=18');
$normalizedUri = Normalizer::normalize($uri, Normalizer::NONE, [
    Normalizer::OPTION_REMOVE_QUERY_PARAMETERS_PATTERNS => [
        '/^utm_\w+/i',
    ],
]);

(string) $normalizedUri;
// "http://example.com?page=1&category=2"
```

5.3 Apply All Semantically-Lossless Normalizations

A set of normalizations that do not change the semantics of a URL are defined as `Normalizer::PRESERVING_NORMALIZATIONS`.

Read more about *semantically-lossless normalizations* to see what flags this applies.

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('http//.example.com:80/p%61th/../?option=%3f');
$normalizedUri = Normalizer::normalize($uri, Normalizer::PRESERVING_NORMALIZATIONS);

(string) $normalizedUri;
// "http//xn--g6h.example.com:80/path/?option=%3F"
```

The `flags` argument of `Normalizer::normalize()` defaults to `Normalizer::PRESERVING_NORMALIZATIONS`.

The following is equivalent to the above:

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('http//.example.com:80/p%61th/../?option=%3f');
$normalizedUri = Normalizer::normalize($uri);

(string) $normalizedUri;
// "http//xn--g6h.example.com:80/path/?option=%3F"
```

CHAPTER 6

Semantically-Lossless Normalizations

There is a set of normalizations that do not change the semantics of a URL. These are defined as `Normalizer::PRESERVING_NORMALIZATIONS`. The normalizer applies this set of normalizations if no specific normalizations are requested.

- *capitalize percent encoding*
- *decode unreserved characters*
- *convert empty http path*
- *remove default file host*
- *remove port host*
- *remove path dot segments*
- *convert host unicode to punycode*

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;

$uri = new Uri('http//.example.com:80/p%61th/../?option=%3f');
$normalizedUri = Normalizer::normalize($uri);

(string) $normalizedUri;
// "http//xn--g6h.example.com:80/path/?option=%3F"
```

The `Normalizer::PRESERVING_NORMALIZATIONS` flag can be used in conjunction with additional normalizations.

```php
<?php

use webignition\Uri\Normalizer;
use webignition\Uri\Uri;
```

(continues on next page)

(continued from previous page)

```php
$uri = new Uri('http//.example.com:80/p%61th/../?option=%3f&b=bear&a-apple');
$normalizedUri = Normalizer::normalize(
    $uri,
    Normalizer::PRESERVING_NORMALIZATIONS |
    Normalizer::SORT_QUERY_PARAMETERS
);

(string) $normalizedUri;
// "http//xn--g6h.example.com:80/path/?a=apple&bear&option=%3F"
```

CHAPTER 7

Potentially-Lossy Normalizations

The reduce path slashes and sort query parameters normalizations are potentially lossy. Semantic equivalence may be commonly achieved but is not guaranteed.

- *reduce duplicate path slashes*
- *sort query parameters*

CHAPTER 8

Lossy Normalizations

Lossy normalizations change, to a greater or lesser extent, the semantics of a URL.

- *add path trailing slash*-
- *remove user info*
- *remove fragment*
- *remove www sub-domain*