Totem Documentation

Release 1.4.0

Baptiste "Talus" Clavié

July 23, 2014

Contents

Changeset calculator and handler for any type of complex data

Requires at least *PHP 5.4*. Compatible PHP 5.5, PHP 5.6, and HHVM

CHAPTER 1

Reference Guide

1.1 Totem : Basic Usage

Totem is a PHP 5.4 library helping you to calculate changes brought to you data. Sure, that does help much to know what it does... Let me clarify then.

Let's say you have a simple array, which looks like this

```
['foo' => 'bar',
 'baz' => ['fubar', 'fubaz']]
```

And then, for some reasons, you need to alter the `foo` key, and instead of having the "bar" value, it should have the "qux" value. So you are stuck with

```
['foo' => 'qux',
 'baz' => ['fubar', 'fubaz']]
```

... And then, let's say you want to say to the end-user "hey, something has changed !". How do you know which key has changed, without actually knowing it (think data edition through a form) ? The answer is simple, it's Totem !

1.1.1 First step : Taking a bunch of Snapshots

Before you modify the data, you'll first need to take a `Snapshot`, which will freeze its data a a given time (all of it). To do that, you just need to instanciate a `Totem\Snapshot` object... Or, to be more precise, it depends on the root type of your data. Currently, here are the built-in types :

- `Totem\Snapshot\ArraySnapshot`, if your root data is an array
- `Totem\Snapshot\ObjectSnapshot`, if your root data is an object.
- `Totem\Snapshot\CollectionSnapshot`, if your root data is a collection.

Warning: The Collection Snapshot is not a recursive snapshot. It will snapshots its arrays and objects, but not its collections. It will consider them as arrays, as there is no easy way to determine what is a collection (except on userland, like the root of the data), and what is the primary key of each elements in said collection.

1.1.2 Second Step : Calculate the diff between your two snapshots

Once you have at least two snapshots (one before your modification, the other one after that modification), you may calculte the diff

```
// let's consider you have two snapshots : one "before" and one "after" the
// modifications
$set = $snapshot['before']->diff($snapshot['after']);
```

You have then a `Totem\Set` object, which is already computed. It is in fact a sort of a container, in which will be stored all the modifications that happened since the "before" snapshot until the "after" snapshot. Each items of this set may have two forms :

- a `Totem\Set` if the item was a snapshot material in the "before" snapshot and in the "after" snapshot ;
- a `Totem\AbstractChange` object if your data was completely changed (if it is a whole different array, object, or whatever else – string, integer, boolean, you name it). This change can be represented in 3 states :
  - a `Totem\Change\Addition` if the key was **added** in the new state ;
  - a `Totem\Change\Modification` if the key was **modified** ;
  - a `Totem\Change\Removal` if the key was **removed** from the data

1.1.3 Third and final step : manipulate your diff

Once you have your `Totem\Set` object, you have an access to each key that was modified ; if it is another `Totem\Set` object, it means that a recursive changeset was created ; if it is only a `Totem\AbstractChange`, you have access to the "old" value (what it was in the old snapshot), and the new value (what is is now)

```
// let's consider you stored the result set in a ``$set`` variable
if ($set->hasChanged('foo')) {
    $change = $set->getChange('foo');
    var_dump($change->getOld(), $change->getNew()); // should dump "bar", "qux"
}
```

There, you have a fully functionnal changeset !