

Zend_Microformat - Pádraic Brady

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Zend_Microformat Component Proposal

Proposed Component Name	Zend_Microformat
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Microformat
Proposers	Pádraic Brady Matthew Weier O'Phinney (Zend Liaison)
Revision	1.0: Pending Community Review (wiki revision: 8)

Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

1. Overview

Designed for humans first and machines second, microformats are a set of simple, open data formats built upon existing and widely adopted standards. Instead of throwing away what works today, microformats intend to solve simpler problems first by adapting to current behaviors and usage patterns (e.g. XHTML, blogging).

Zend_Microformat will form the backbone of a set of both compound and elemental microformats implemented and proposed separately (so anyone else can take part in the fun!). The initial microformat targeted for implementation is the XHTML Friends Network (XFN) microformat. Should this proposal be accepted into the Zend Framework, subsequent proposals will be made for many others.

The purpose of the core Zend_Microformat component is to act as an overall URL fetcher and parser which targets any desired microformat to be parsed from the resulting HTML/XHTML and returned in a more specific collection object per microformat. The same will be achievable using the sub-components for each microformat, but having a central utility class just offers more consistency to the overall API.

2. References

- [About Microformats](#)
- [Microformats.org Wiki](#)
- [Zend_Microformat_Hcard Proposal](#)

3. Component Requirements, Constraints, and Acceptance Criteria

These requirements are subject to modification pending community review and my own personal whims (which is what you get for reading something marked Under Construction 😊).

- This component **will** provide central responsibility for fetching URLs, and managing the parsing and formation of collection objects, for the selected microformat.
- This component **will** be used to enforce a relatively common API across all sub-components. Reasonably, because each microformat has it's own unique requirements.
- This component **may** act as an anchor point for a microformat factory method.
- In general, this component (or its sub-components) **will** allow for both the parsing of, and generation of, microformat XHTML, HTML and XML.
- Where relevant, this component **will** delegate to other Zend Framework components should it prove a path of least resistance.

4. Dependencies on Other Framework Components

- Zend_Http
- Zend_Exception

5. Theory of Operation

The core Zend_Microformat component is not intended to prove a huge overarching presence. Instead it's a focal point for common functionality required for all microformats. A simple operation would involve using Zend_Microformat to fetch (using Zend_Http) a URL, cache the resulting XHTML/HTML/XML, and subsequently parse the cached output for one or more collections of microformat data.

Where sub-components for specific microformats are utilised, Zend_Microformat will be utilised via composition. There is no intent for Zend_Microformat to include static methods which enable Zend_Http instance reuse since URL results are temporarily cached for any subsequent operations.

6. Milestones / Tasks

- Milestone 1: Assemble some initial use cases to explore desired API
- Milestone 2: Commence proof-of-concept coding (will discard after)
- Milestone 3: Pending review, complete initial development using TDD
- Milestone 4: Complete acceptance testing, verify unit test coverage
- Milestone 5: Let's assume documentation is written during development

7. Class Index

- Zend_Microformat
- Zend_Microformat_Exception

Please see Zend_Microformat sub-proposals for additional classes in this namespace.

8. Use Cases

UC-01

Fetch a result object containing all supported Microformat data possible

```
$mf = new Zend_Microformat('http://www.example.com');  
$results = $mf->findAll();
```

UC-02

Fetch a result object containing only XFN data

```
$mf = new Zend_Microformat('http://www.example.com');
$results = $mf->find('xfn');
```

UC-03

Fetch a result object containing only XFN data, and try again for hCard data

```
$mf = new Zend_Microformat('http://www.example.com');
$xfnResults = $mf->find('xfn');
$hcardResults = $mf->find('hcard');
```

UC-04

Iterate across hCard results to list FN values

```
$mf = new Zend_Microformat('http://www.example.com');
$hcards = $mf->find('hcard');
echo 'Located ', count($hcards), ' hCard(s):', PHP_EOL;
foreach($hcards as $hc) {
    echo $hc->fn, "\n";
}
```

UC-05

Iterate across hCard results to list N values*

```
$mf = new Zend_Microformat('http://www.example.com');
$hcards = $mf->find('hcard');
echo 'Located ', count($hcards), ' hCard(s):', PHP_EOL;
foreach($hcards as $hc) {
    echo $hc->n->nickname, PHP_EOL;
    echo $hc->n->url, PHP_EOL;
    echo $hc->n->tel, PHP_EOL, PHP_EOL;
}
```

UC-06

Create a new hCard, and generate HTML markup for embedding

Note that hCard FN and N properties are required under the hCard specification.

```
$mf = new Zend_Microformat;
$hcard = $mf->new('hcard');
$hcard->fn = 'Joe Bloggs';
$hcard->n->nickname = 'jbloggs';
$hcard->n->honorific-prefix = 'Mr';
$hcard->n->given-name = 'Joe';
$hcard->n->family-name = 'Bloggs';
echo $hcard->generate();
```

UC-07

Create a new hCard, using Zend_Config

jbloggs.xml

```
<?xml version="1.0"?>
<configdata>
  <hcard>
    <fn>Joe Bloggs</fn>
    <n>
      <nickname>jbloggs</nickname>
      <honorific-prefix>Mr.</honorific-prefix>
      <given-name>Joe</given-name>
      <family-name>Bloggs</family-name>
    </n>
  </hcard>
</configdata>
```

```
$config = new Zend_Config_Xml('./jbloggs.xml', 'hcard');

$mf = new Zend_Microformat;
$hcard = $mf->new('hcard', $config);
echo $hcard->generate();
```

9. Class Skeletons

Please see the above use cases. Based on these, source code will be developed following Test-Driven Development and therefore class skeletons will not be presented in this proposal prematurely. The TDD process will implement Zend_Microformat specifically to enable the use cases presented above.

```
]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>
```