Zend_Tool_Pattern - In need of proposer

<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_Tool_Pattern Component Proposal

Proposed Component Name	Zend_Tool_Pattern
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Tool_Pattern
Proposers	Missing
Zend Liaison	TBD
Revision	0.1 - 4 June 2009: Initial Draft. 0.5 - 6 June 2009: Initial Draft. (wiki revision: 11)

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

Zend_Tool_Pattern will provide some common and simple design patterns that are used frequently in the development of your application. Think accessors/mutators and a singleton. In addition, it will provide a way to easily implement interfaces by adding the specified interface to the includes list and generating the method skeletons.

2. References

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** provide support for generating simple design patterns within a single class.
- This component **will** provide support for generating methods provided by interfaces.
- This component **will not** provide support for design patterns spanning multiple classes (yet ;).

4. Dependencies on Other Framework Components

- Zend_CodeGenerator
- Zend_Tool_Framework

5. Theory of Operation

These providers for Zend_Tool will allow you to take an existing class and apply simple design patterns and interfaces to it. For example:

Accessor
Create or remove an accessor for the specified property in the specified class. If not present, it will create the correspoding (protected) property. It will follow the "get" naming convention.
Provider name: accessor
Available actions: create
Parameters: create [name, class]
CLI usage: zf create accessor foo MyClass
Generates:

```
protected $_foo = null;

public function getFoo()
{
    return $this->_foo;
}
```

Mutator
Create or remove a mutator for the specified property in the specified class. If not present, it will create the correspoding (protected) property. It will follow the "set" naming convention.
Provider name: mutator
Available actions: create
Parameters: create [name, class, returnThis=true]
CLI usage: zf create mutator foo MyClass
Generates:

```
protected $_foo = null;

public function setFoo($foo)
{
    $this->_foo = $foo;

    return $this;
}
```

Interfaces
This will generate method skeletons for the specified interface and will add that interface to the class declaration. It will use reflection to determine the methods provided by the interface to determine what to generate, making this provider useable for both all interfaces.
Provider name: interface
Available actions: implement
Parameters: implement [name, class]
CLI usage: zf implement countable MyClass
Generates:

```
class MyClass implements Countable
{
    public function count()
    {
    }
}
```

Singleton
Will turn the specified class into a singleton with by default a protected constructor and a static getInstance() method. If a public constructor is already present it will convert it to a protected constructor. If there's already a private constructor it will be left as-is, unless visibility is explictly set to protected.
Provider name: singleton
Available actions: create
Parameters: create [class, name=getInstance, visibility=protected]
CLI usage: zf create singleton MyClass
Generates:

```
protected static $_instance = null;

public static function getInstance()
{
    if (self::$_instance === null) {
        self::$_instance = new self();
    }

    return self::$_instance;
}

protected function __construct()
{
}
```

6. Milestones / Tasks

- Milestone 1: Collect feedback on this proposal
- Milestone 2: Working prototype checked into the incubator
- Milestone 3: Unit tests exist, work, and are checked into SVN.
- Milestone 4: Initial documentation exists.
- Milestone 5: Promotion to trunk

7. Class Index

- Zend_Tool_Pattern_Accessor
- Zend_Tool_Pattern_Mutator
- Zend_Tool_Pattern_Interface
- Zend_Tool_Pattern_Singleton

8. Use Cases

9. Class Skeletons

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

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