Zend_CommonInterface domain

<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_CommonInterface Component Proposal

Proposed Component Name	Zend_CommonInterface
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_CommonInterface
Proposers	Ralph Schindler
Revision	1.1 - 8 November 2006: Updated from community comments. (wiki revision: 6)

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_CommonInterface is a domain that will server to host common interfaces that framework components can implement to insure proper coupling without using simply convention or guesswork.

2. References

- Interfaces
- Countable

3. Component Requirements, Constraints, and Acceptance Criteria

This domain allows compontents to implement common intefaces to ensure that when optional coupling of components is a nessessity, that certain method expectations can be met.

Consider the following example: Zend_Auth needs to be able to use a persistent storage module for keeping identity stored accross multiple requests. One option for the web arena is Zend_Session. But, perhaps the developer wants to implenent this in a file, or in the registry for a desktop app or somethign similar. Or, the devloper wants to use some kind of central server and home grown cookie implementation ot handle his persistent storage needs. In this case, Zend_Auth would need to use some kind module that implements a common interface.. In this particular example, the common interface would be a "Container" or a "Persistent Storage Container".

As we start drilling down into more specific components as they relate to the MVC paridigm, common interfaces are going become more important if we want to keep the level of coupling between modules low.

4. Dependencies on Other Framework Components

- none

5. Theory of Operation

They are only interfaces

6. Milestones / Tasks

- Milestone 1: design notes will be published here
- Milestone 2: ?
- Milestone 3: ?

7. Class Index

- Zend_CommonInterface_Container

8. Use Cases

UC-01

```
class Zend_Session implements Zend_CommonInterface_Container { ... }

Zend_Auth_Session extends Zend_Session { ... }


// by default Zend_Auth would use Zend_Auth_Session, but to illustrate:
Zend_Auth::setStorageModule(new Zend_Auth_Session());

   // in Zend_Auth
   ...
   public static function setStorageContainer(Zend_CommonInterface_Container) { ... }
   ...


// now the developer can optionally use another module for storage, perhaps My_HashObject which
implements Zend_CommonInterface_Container
Zend_Auth::setStorageModule(new My_HashObject(...));
```

9. Class Skeletons

```
interface Zend_CommonInterface_Container
{
    public function __get($name);
    public function __set($name, $value);
    public function __isset($name);
    public function __unset($name);
}
```

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