Zend_Json_Expr to allow Javascript Expressions (functions) to be encoded using Zend_Json

<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_Json_Expr Component Proposal

Proposed Component Name	Zend_Json_Expr
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Json_Expr
Proposers	Oscar Reales
Zend Liaison	Matthew Weier O'Phinney
Revision	1.0 - 10 August 2008: Initial Draft. (wiki revision: 13)

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

Add the possibility of including functions and native javascripts expressions to be encoded by Zend_Json, generating right syntax in the outputted json. The main idea is follow the Zend_Db_Expr pattern, so to include a Javascript Expression to be encoded in an object/array, will be enough to uses something similar to Zend_Json_Expr('javascript expression').

2. References

- Based on this trick to sending functions over json

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** allow include Functions and Javascript Expressions in arrays/object to be encoded with Zend_Json
- This component **will** require the creation of a new Zend_Json_Expr class, very similar to Zend_Db_Expr
- This component **will** need to re-code the Zend_Json encoding proccess and includes 2 private properties to the class that will hold some values temporarily during the encoding proccess

- This component **will** be functional with Array/Objects to be encoded. Encoding from XML will need to be handle maybe in other way

4. Dependencies on Other Framework Components

- Zend_Json

5. Theory of Operation

To include a native Javascript Expression in a PHP Array/Object will be required to use Zend_Json_Expr class, with the following syntax: Zend_Json_Expr('javascript expression').

This Zend_Json_Expr class will be used to find and temporarily replace pre-encoding proccess all the javascript expressions we want to encode.

The value to be encoded in json format will be pre-encoding parsed and all finded Zend_Json_Expr will be temporarily keeped in an array. Each matched Zend_Json_Expr will be replaced by a single id. In this way the encoding can be runned without errors. After encoding proccess finish, it is required to parse the result in the inverse way: replacing the single temporary "ids" by their corresponding original Javascript Expressions.

Because the result it is already a String after the encoding, it is possible to includes the original Javascript Expressions without """.

6. Milestones / Tasks

- Milestone 1: [DONE] Create the classes skeleton
- Milestone 2: [DONE] Working prototype
- Milestone 3: [DONE] Review code to follow ZF coding conventions
- Milestone 4: Create - Modify Unit Testing
- Milestone 5: Documentation

7. Class Index

- Zend_Json_Expr
- Zend_Json

8. Use Cases

UC-01

This is a sample of using the Zend_Json_Expr class in a controller that will returns a json encoded object with one simple javascript function:

```
<?php
class JsonUseCase extends Zend_Controller_Action
{

function indexAction()
{
Zend_Loader::loadClass('Zend_Json_Expr');
```

```
$encodedResult = array(
'integer'=>9,
'string'=>'test string',
'function'=>Zend_Json_Expr('function(){window.alert("javascript function encoded by Zend_Json")}')
);


$this->_helper->json->sendJson($resultado);
//it will returns json encoded string:
//{"integer":9,"string":"test string","function":function(){window.alert("javascript function encoded by Zend_Json")}}
}


}
```

9. Class Skeletons

```
<?php

/**
 * This is an object to hold temporarily a Javascript Expression (string)
 * during the encoding proccess.
 *
 */

class Zend_Json_Expr
{
    /**
     * Storage for javascript expression.
     *
     * @var string
     */
    protected $_expression;

    /**
     * Constructor
     *
     * @param string $expression the expression to hold.
     */
    public function __construct($expression)
    {
        $this->_expression = (string) $expression;
    }

    /**
     * @return string holded javascript expression.
     */
    public function __toString()
    {
        return $this->_expression;
    }

}




<?php

/**
 * Rewriting the actual Zend_Json to accomodate the Zend_Json_Expr proposal
 *
 */

class Zend_Json {
```

```
/**
 * Array where the javascriptExpressions will be keeped during conversion
 *
 * @var array of strings with javascripts expressions
 */
private static $_javascriptExpressions = array();

/**
 * Array where the replacing keys will be keeped during conversion.
 *
 * After encoding we will use this keys to inject original javascript
 * expressions in the final encoded string
 *
 * @var array
 */
private static $_replacingKeys = array();


/**
 * Rewriting of actual encode method
 *
 * @param mixed $valueToEncode
 * @param boolean $cycleCheck Optional;
 * whether or not to check for object recursion; off by default
 * @return string JSON encoded object
 */
public static function encode($valueToEncode, $cycleCheck = false)
{

    //pre-encoding look for Zend_Json_Expr objects and replacing by ids
    self::_checkForExpressions($valueToEncode);


    //encoding
    $encodedResult = '';
    if (function_exists('json_encode')
        && self::$useBuiltinEncoderDecoder !== true) {
            $encodedResult = json_encode($valueToEncode);
    }else{
       require_once 'Zend/Json/Encoder.php';
       $encodedResult = Zend_Json_Encoder::encode($valueToEncode, $cycleCheck);
    }


    //post-encoding replacing the "ids" by their
    //corresponding original javascript expressions
    $finalEncodedString = str_replace(
                self::$_replacingKeys,
                self::$_javascriptExpressions,
                $encodedResult);

    return $finalEncodedString;
}

/**
* Parse the value to encode and replace in arrays the javascript expressions
* in order to re-write them in the
* final result, after the encoding to json has been finished
*
* @param mixed $valueToCheck the string/expression to check
*/
private static function _checkForExpressions(&$valueToCheck){
    foreach($valueToCheck as $key=>&$value){
        if(is_a($value, 'Zend_Json_Expr')){
            $index = count(self::$_javascriptExpressions);
            self::$_javascriptExpressions[] = $value->__toString();
            self::$_replacingKeys[] = '"%'.($index).'%"';
            $value = '%'.($index).'%';
        }elseif(is_array($value) || is_object($value)){
            self::_checkForExpressions($value);
```

```
}
```

```
            }
        }
    }
```

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