Zend_Log factory() - Martin Roest

<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_Log Component Proposal

Proposed Component Name	Zend_Log
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Log
Proposers	Martin Roest
Zend Liaison	Matthew Weier O'Phinney
Revision	1.0 - 29 september 2008: Initial Draft. (wiki revision: 14)

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

This proposition covers a new factory method in Zend_Log that eases the creation and configuration of the Zend_Log object.

2. References

3. Component Requirements, Constraints, and Acceptance Criteria

4. Dependencies on Other Framework Components

- zend_Log_Writer_*
- Zend_Log_Filter_*

5. Theory of Operation

Enable the creation and setup of the Zend_Log object from a Zend_Config object similar the Zend_Db::factory()

A user is able to setup a configuration file with the help of Zend_Config creates the config object and passes the relevant section to the Zend_Log::factory(). The factory method will create the Zend_Log object and adds the configured and enabled log writers.

This allows easy configuration and enable/disable logwriters per config section.

6. Milestones / Tasks

- Milestone 1: Change the log writers to accept a Zend_Config object or Array when constructing the writer
- Milestone 2: Add the factory method

7. Class Index

- Zend_Log
- Zend_Log_Writer_*
- Zend_Log_Filter_*
- Zend_Log_Exception

8. Use Cases

UC-01

config.ini (The namespace property is only to indicate support for namespace prefix. It defaults to Zend_Log_Writer):

```
log.file.writerName = "Stream"
log.file.writerNamespace = "Zend_Log_Writer"
log.file.writerParams.stream = "/path/to/application.log"
log.file.filterName = "Priority"
log.file.filterParams.priority = "Zend_Log::CRIT"
log.file.filterParams.operator = "<="
log.firebug.writerName = "Firebug"
log.db.writerName = "Db"
log.db.writerParams.adapter  "Mysqli"
log.db.writerParams.params.host = "localhost"
log.db.writerParams.params.username = "user"
log.db.writerParams.params.password = "pass"
log.db.writerParams.params.dbname = "dbname"
```

bootstrap/setup

```
require 'Zend/Log.php';
$logger = Zend_Log::factory($config->log);
```

9. Class Skeletons

Zend_Log

```
/**
     * Factory to construct the Zend_Log and add writers based on the
     * configuration
     *
     * $config can be an array of an instance of Zend_Config
     *
     * @param mixed $config Array or instance of Zend_Config
     * @return Zend_Log
     */
    public static function factory($config = array())
    {
        if($config instanceof Zend_Config) {
            $config = $config->toArray();
        }

        if(!is_array($config)) {
            /** @see Zend_Log_Exception */
            require_once 'Zend/Log/Exception.php';
            throw new Zend_Log_Exception('Writer configuration must be an array or instance of
Zend_Config!');
        }

        /** Do we have one or more writers configured? **/
        if(!is_array(current($config))) {
            $config = array($config);
        }

        $logger = new Zend_Log();

        foreach($config as $writer) {

            $writerName = self::getFullClassName($writer['writerName'], 'writerNamespace', $writer,
'Zend_Log_Writer');

            /*
             * Load the writer class.  This throws an exception
             * if the specified class cannot be loaded.
             */
            @Zend_Loader::loadClass($writerName);

            $writerConfig = (isset($writer['writerParams'])) ? $writer['writerParams'] : array();
            $writerObj = new $writerName($writerConfig);

            if(! $writerObj instanceof Zend_Log_Writer_Abstract) {
                /**
                 * @see Zend_Log_Exception
                 */
                require_once 'Zend/Log/Exception.php';
                throw new Zend_Log_Exception("Writer class '$writerName' does not extend
Zend_Log_Writer_Abstract");
            }

            if(isset($writer['filterName'])) {
                $filterClassName = self::getFullClassName($writer['filterName'], 'filterNamespace',
$writer , 'Zend_Log_Filter');

                @Zend_Loader::loadClass($filterClassName);

                $filterConfig = (isset($writer['filterParams'])) ? $writer['filterParams'] : array();
                $filterObj = new $filterClassName($filterConfig);

                if(! $filterObj instanceof Zend_Log_Filter_Interface) {
                    /**
```

```
                     * @see Zend_Db_Exception
                     */
                    require_once 'Zend/Log/Exception.php';
                    throw new Zend_Log_Exception("Filter class '$filterClassName' does not extend
Zend_Log_Filter_Interface");
                }

                $writerObj->addFilter($filterObj);
            }

            $logger->addWriter($writerObj);
        }
        return $logger;
    }

    /**
     * Get the full class name which can be used by the Zend_Loader
     *
     * @param string classname
     * @param string namespace to search for
     * @param array array with parameters
     * @param string default namespace
     */
    protected static function getFullClassName($className, $needle, Array $haystack,
$defaultNamespace)
    {
        if(isset($haystack[$needle])) {
           $defaultNamespace = $haystack[$needle];
        }

        $fullClassName = strtolower($defaultNamespace . '_' . $className);
```

```
        $fullClassName = str_replace(' ', '_', ucwords(str_replace('_', ' ', $fullClassName)));
        return $fullClassName;
    }
```

Example of how to implement it with the Zend_Log_Writer_Db

```
/**
     * Class constructor
     *
     * @param mixed $dbOrConfig   Zend_Db_Adapter Database adapter instance or config array
     * @param string $table         Log table in database
     * @param array $columnMap
     */
    public function __construct($dbOrConfig, $table = null, $columnMap = null)
    {
        if(count(func_get_args()) == 1) {

            if(!is_array($dbOrConfig)) {
                throw new Zend_Log_Exception("Unsupported configuration. Pass a configuration array");
            }

            $this->_table = $dbOrConfig['table'];
            unset($dbOrConfig['table']);

            $this->_columnMap = $dbOrConfig['columnMap'];
            unset($dbOrConfig['columnMap']);

            /** Construct Zend_Db **/
            require_once 'Zend/Db.php';
            $this->_db = Zend_Db::factory($dbOrConfig);

        } else {
            $this->_db    = $dbOrConfig;
            $this->_table = $table;
            $this->_columnMap = $columnMap;
        }
    }
```

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