Zend_Log_Writer_Syslog - Thomas Gelf

<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_Writer_Syslog Component Proposal

Proposed Component Name	Zend_Log_Writer_Syslog
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Log_Writer_Syslog
Proposers	Thomas Gelf
Zend Liaison	Matthew Weier O'Phinney
Revision	1.0 - 19 September 2008: Posted to mailing list, created proposal after getting some feedback 1.1 - 24 September 2008: Removed Exceptions as PHP doesn't behave as documentation says 1.2 - 27 September 2008: Added "Multi-Instance-Capability" (wiki revision: 21)

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_Log_Writer_Syslog implements a syslog writer for Zend_Log.

2. References

3. Component Requirements, Constraints, and Acceptance Criteria

4. Dependencies on Other Framework Components

- Zend_Log_Writer_Abstract

5. Theory of Operation

This component implements a syslog writer for Zend_Log.

Syslog benefits:

- There are lots of tools taking care of logfile rotation and such jobs
- You can scale out your application and easily centralize your logs
- Fast, small overhead

6. Milestones / Tasks

- Milestone 1: [DONE] Initial class design
- Milestone 2: [DONE] Create prototype
- Milestone 3: [DONE] Finish proposal and submit for community review
- Milestone 4: Create unit tests

7. Class Index

- Zend_Log_Writer_Syslog

8. Use Cases

UC-01

```
// will log to syslog as application "Zend_Log" with facility LOG_USER
$logger = new Zend_Log();
$logger->addWriter(new Zend_Log_Writer_Syslog());
```

UC-02

```
// will log to syslog as application "MyApplication" with facility LOG_USER
$logger = new Zend_Log();
$writer = new Zend_Log_Writer_Syslog('MyApplication');
$logger->addWriter($writer);
```

UC-03

```
// will log to syslog as application "MyApplication" with facility LOG_LOCAL0
// NB: LOCAL0-7 are not available on Windows systems
$logger = new Zend_Log();
$writer = new Zend_Log_Writer_Syslog('MyApplication', LOG_LOCAL0);
$logger->addWriter($writer);
```

UC-04

```
// will log to syslog as application "MyApplication" with facility LOG_USER
// critical errors are also logged as "NotMyApplication", facility LOG_LOCAL0
// openlog() will automagically be called each time an application name or
// facility change really needs to take place
$logger = new Zend_Log();
$writer1 = new Zend_Log_Writer_Syslog('MyApplication');
$writer2 = new Zend_Log_Writer_Syslog('NotMyApplication');
$filter = new Zend_Log_Filter_Priority(Zend_Log::CRIT);
$writer2->addFilter($filter);
$logger->addWriter($writer1);
$logger->addWriter($writer2);
```

9. Class Skeletons

```
<?php
/**
 * Zend Framework
 *
 * LICENSE
 *
 * This source file is subject to the new BSD license that is bundled
 * with this package in the file LICENSE.txt.
 * It is also available through the world-wide-web at this URL:
 * http://framework.zend.com/license/new-bsd
 * If you did not receive a copy of the license and are unable to
 * obtain it through the world-wide-web, please send an email
 * to license@zend.com so we can send you a copy immediately.
 *
 * @category   Zend
 * @package    Zend_Log
 * @subpackage Writer
 * @copyright  Copyright (c) 2005-2008 Zend Technologies USA Inc. (http://www.zend.com)
 * @license    http://framework.zend.com/license/new-bsd     New BSD License
 */

/** Zend_Log_Writer_Abstract */
require_once 'Zend/Log/Writer/Abstract.php';

/**
 * Writes log messages to syslog
 *
 * @category   Zend
 * @package    Zend_Log
 * @subpackage Writer
 * @copyright  Copyright (c) 2005-2008 Zend Technologies USA Inc. (http://www.zend.com)
 * @license    http://framework.zend.com/license/new-bsd     New BSD License
 */
class Zend_Log_Writer_Syslog extends Zend_Log_Writer_Abstract
{
    /**
     * Maps Zend_Log priorities to PHP's syslog priorities
     * @var array
     */
    protected $_priorities = array(Zend_Log::EMERG  => LOG_EMERG,
                                   Zend_Log::ALERT  => LOG_ALERT,
                                   Zend_Log::CRIT   => LOG_CRIT,
                                   Zend_Log::ERR    => LOG_ERR,
                                   Zend_Log::WARN   => LOG_WARNING,
                                   Zend_Log::NOTICE => LOG_NOTICE,
                                   Zend_Log::INFO   => LOG_INFO,
                                   Zend_Log::DEBUG  => LOG_DEBUG);
```

```
    /**
     * The default log priority - for unmapped custom priorities
     * @var string
     */
    protected $_defaultPriority = LOG_NOTICE;

    /**
     * Last application name set by a syslog-writer instance
     * @var string
     */
    protected static $_lastApplication;

    /**
     * Last facility name set by a syslog-writer instance
     * @var string
     */
    protected static $_lastFacility;

    /**
     * Application name used by this syslog-writer instance
     * @var string
     */
    protected $_application;

    /**
     * Facility used by this syslog-writer instance
     * @var string
     */
    protected $_facility;

    /**
     * Class constructor
     */
    public function __construct($application = 'Zend_Log', $facility = LOG_USER)
    {
        $this->_application = $application;
        $this->_facility = $facility;
        $this->initializeSyslog();
    }

    /**
     * Initialize syslog / set application name and facility
     *
     * @param  string  $application   application name
     * @param  string  $facility      syslog facility
     * @return void
     */
    protected function initializeSyslog()
    {
        self::$_lastApplication = $this->_application;
        self::$_lastFacility = $this->_facility;
        openlog($this->_application, LOG_PID, $this->_facility);
    }

    /**
     * Set syslog facility
     *
     * @param  string  $facility      syslog facility
     * @return void
     */
    public function setFacility($facility)
    {
        if ($this->_facility === $facility) return;
        $this->_facility = $facility;
        $this->initializeSyslog();
    }

    /**
     * Set application name
```

```
     *
     * @param  string  $application   application name
     * @return void
     */
    public function setApplicationName($application)
    {
        if ($this->_application === $application) return;
        $this->_application = $application;
        $this->initializeSyslog();
    }

    /**
     * Close syslog.
     *
     * @return void
     */
    public function shutdown()
    {
        closelog();
    }

    /**
     * Write a message to syslog.
     *
     * @param  array  $event  event data
     * @return void
     */
    protected function _write($event)
    {
        if (array_key_exists($event['priority'], $this->_priorities)) {
            $priority = $this->_priorities[$event['priority']];
        } else {
            $priority = $this->_defaultPriority;
        }

        if ($this->_application !== self::_lastApplication
            || $this->_facility !== self::_lastFacility)
        {
            $this->initializeSyslog();
        }
```

```
            syslog($priority, $event['message']);
        }
    }
```

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

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