Zend_Growl + Zend_Log_Writer_Growl - Tom Graham

<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_Growl + Zend_Log_Writer_Growl Component Proposal

Proposed Component Name	Zend_Growl + Zend_Log_Writer_Growl
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Growl + Zend_Log_Writer_Growl
Proposers	Tom Graham
Zend Liaison	Matthew Weier O'Phinney
Revision	1.1 - 21 September 2008: Initial Draft. (wiki revision: 9)

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_Growl is a simple component for sending Growl notifications.
Zend_Log_Writer_Growl utilises Zend_Growl to send Growl notifications upon log events.

2. References

- Growl network protocol
- Noginn_Growl implementation
- Noginn_Log_Writer_Growl implementation
- Blog post about the component

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** send network Growl notifications via UDP
- This component **will** provide a log writer that sends Growl notifcations

4. Dependencies on Other Framework Components

- Zend_Log_Writer_Abstract
- Zend_Log_Filter_Priority
- Zend_Log_Exception
- Zend_Exception

5. Theory of Operation

6. Milestones / Tasks

- Milestone 1: Design
- Milestone 2: Working prototype checked into the incubator supporting use cases
- Milestone 4: Unit tests exist, work, and are checked into SVN.
- Milestone 5: Initial documentation exists.

7. Class Index

- Zend_Growl
- Zend_Log_Writer_Growl

8. Use Cases

UC-01

```
$growl = new Zend_Growl('127.0.0.1', 'password');
$growl->addNotification('Messages');
$growl->notify('Messages', 'Title', 'Message description', Noginn_Growl::PRIORITY_LOW, true);
```

UC-02

```
$growl = new Zend_Growl('127.0.0.1', 'password');
$writer = new Zend_Log_Writer_Growl($growl);
$logger = new Zend_Log($writer);

// Sends Growl notification with the Title 'Emergency'
$logger->emerg('Log message');
```

9. Class Skeletons

```
||Zend_Growl||
<?php
class Zend_Growl
{
    /**
     * The priority flags
     */
    const PRIORITY_LOW = -2;
    const PRIORITY_MODERATE = -1;
    const PRIORITY_NORMAL = 0;
    const PRIORITY_HIGH = 1;
    const PRIORITY_EMERGENCY = 2;

    /**
     * The types of packets
     */
    const TYPE_REGISTRATION = 0;
    const TYPE_NOTIFICATION = 1;

    /**
     * The growl network protocol version
     */
    const PROTOCOL_VERSION = 1;

    /**
     * The port to use when sending the UDP packet
     */
    const UDP_PORT = 9887;

    /**
     * The IP address of the growl server
     *
     * @var string
     */
    public $address;

    /**
     * The password of the growl server
     *
     * @var string
     */
    public $password;

    /**
     * The application name to register with Growl
     *
     * @var string
     */
    public $application = 'Noginn_Growl';

    /**
     * The types of notifications that may be sent
     *
     * @var array
     */
    public $notifications = array();

    /**
     * Whether the application has already been registered
     *
     * @var bool
     */
    protected $_registered = false;

    /**
     * Constructor
     *
     * Usage:
     * <code>
```

```
     * // Setup the Growl client
     * $growl = new Noginn_Growl('127.0.0.1', 'password', 'My Growl App');
     * $growl->addNotification('Alert');
     * $growl->addNotification('Warning', array('enabled' => false));
     *
     * // Send the notifications
     * $growl->notify('Alert', 'Title', 'Message', Zend_Growl::PRIORITY_HIGH, true);
     * $growl->notify('Warning', 'Title', 'Message', Zend_Growl::PRIORITY_LOW, false);
     * </code>
     *
     * @param string $address The IP address of the Growl server
     * @param string $password The password of the Growl server
     * @param string $application The application name to register with the Growl server
     * @return void
     */
    public function __construct($address = null, $password = null, $application = null) {}

    /**
     * Set the IP address and password of the Growl server
     *
     * @param string $address
     * @param string $password
     * @return void
     */
    public function setAddress($address, $password = '') {}

    /**
     * Add a notification supported by the application
     *
     * @param string $name
     * @param bool $enabled
     * @return void
     */
    public function addNotification($name, $options = array()) {}

    /**
     * Send application registration packet
     *
     * @return void
     */
    protected function _register() {}

    /**
     * Send a notification
     *
     * @param string $name The notification name
     * @param string $title The title of the notification
     * @param string $description The description of the notification
     * @param int $priority The priority of the notification
     * @param bool $sticky Whether the notification is sticky or not
     * @return void
     */
    public function notify($name, $title, $description, $priority = self::PRIORITY_NORMAL, $sticky =
false) {}

    /**
     * Send the UDP packet
     *
     * @param string $data
     * @return bool
     */
    public function send($data) {}
}
||Zend_Log_Writer_Growl||
<?php
class Zend_Log_Writer_Growl extends Zend_Log_Writer_Abstract
{
    /**
     * Maps logging priorities to Growl notification priority
     * @var array
```

```
     */
    protected $_priorityTypes = array(
        Zend_Log::EMERG  => Zend_Growl::PRIORITY_EMERGENCY,
        Zend_Log::ALERT  => Zend_Growl::PRIORITY_HIGH,
        Zend_Log::CRIT   => Zend_Growl::PRIORITY_HIGH,
        Zend_Log::ERR    => Zend_Growl::PRIORITY_HIGH,
        Zend_Log::WARN   => Zend_Growl::PRIORITY_NORMAL,
        Zend_Log::NOTICE => Zend_Growl::PRIORITY_MODERATE,
        Zend_Log::INFO   => Zend_Growl::PRIORITY_LOW,
        Zend_Log::DEBUG  => Zend_Growl::PRIORITY_LOW
    );

    /**
     * The default Growl notification priority for un-mapped priorities
     * @var string
     */
    protected $_defaultPriorityType = Noginn_Growl::PRIORITY_NORMAL;

    /**
     * Maps logging priorities to Growl notification title
     * @var array
     */
    protected $_priorityTitles = array(
        Zend_Log::EMERG  => 'Emergency',
        Zend_Log::ALERT  => 'Alert',
        Zend_Log::CRIT   => 'Critical',
        Zend_Log::ERR    => 'Error',
        Zend_Log::WARN   => 'Warning',
        Zend_Log::NOTICE => 'Notice',
        Zend_Log::INFO   => 'Info',
        Zend_Log::DEBUG  => 'Debug'
    );

    /**
     * The default Growl notification title for un-mapped priority
     * @var string
     */
    protected $_defaultPriorityTitle = 'Log';

    /**
     * Flag indicating whether the log writer is enabled
     * @var boolean
     */
    protected $_enabled = true;

    protected $_growl;

    /**
     * Class constructor
     *
     * @param Noginn_Growl $growl The instance of the Growl component to use
     */
    public function __construct(Noginn_Growl $growl)
    {
        $this->setGrowl($growl);

        if (php_sapi_name()=='cli') {
            $this->setEnabled(false);
        }
    }

    public function setGrowl($growl)
    {
        foreach ($this->_priorityTitles as $name) {
            $growl->addNotification($name);
        }

        $this->_growl = $growl;
    }
```

```
    public function getGrowl()
    {
        return $this->_growl;
    }

    /**
     * Enable or disable the log writer.
     *
     * @param boolean $enabled Set to TRUE to enable the log writer
     * @return boolean The previous value.
     */
    public function setEnabled($enabled)
    {
        $previous = $this->_enabled;
        $this->_enabled = $enabled;
        return $previous;
    }

    /**
     * Determine if the log writer is enabled.
     *
     * @return boolean Returns TRUE if the log writer is enabled.
     */
    public function getEnabled()
    {
        return $this->_enabled;
    }

    /**
     * Formatting is not possible on this writer
     *
     * @return void
     */
    public function setFormatter($formatter)
    {
        require_once 'Zend/Log/Exception.php';
        throw new Zend_Log_Exception(get_class() . ' does not support formatting');
    }

    /**
     * Send a Growl notification with the log message
     *
     * @param array $event The event data
     * @return void
     */
    protected function _write($event)
    {
        if (!$this->getEnabled()) {
            return;
        }

        if (array_key_exists($event['priority'],$this->_priorityTypes)) {
            $type = $this->_priorityTypes[$event['priority']];
        } else {
            $type = $this->_defaultPriorityType;
        }

        if (array_key_exists($event['priority'],$this->_priorityTitles)) {
            $title = $this->_priorityTitles[$event['priority']];
        } else {
            $title = $this->_defaultPriorityTitle;
        }

        $growl = $this->getGrowl();
```

```
        $growl->notify($title, $title, $event['message'], $type);
    }
}
```

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