Zend_Scheduler - Matthew Ratzloff

<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_Scheduler Component Proposal

Proposed Component Name	Zend_Scheduler
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Scheduler
Proposers	Matthew Ratzloff
Revision	0.3 - 9 December 2006: Scheduling rules are fully-functional now and are built on Zend_Date. Combined separate Rule_* classes into Zend_Scheduler_Task_Rule. **Subversion: http://framework.zend.com/svn/laboratory/Zend_Scheduler/** (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_Scheduler is a request-based job scheduling component designed to allow easy scheduling of tasks based on simple temporal constraints, without the use of OS-based applications such as Crontab.

2. References

- Crontab - Partial basis of constraint syntax

3. Component Requirements, Constraints, and Acceptance Criteria

- Common task creation interface
- Modular design

4. Dependencies on Other Framework Components

- Zend_Controller_Front
- Zend_Controller_Request_Abstract
- Zend_Controller_Router
- Zend_Exception

5. Theory of Operation

Users create a Zend_Scheduler instance, then add named Zend_Scheduler_Task instances to the scheduler. Multiple requests can be assigned to any given task, and a limit can be placed on the number of tasks to execute in any given request.

6. Milestones / Tasks

7. Class Index

- Zend_Scheduler
- Zend_Scheduler_Exception
- Zend_Scheduler_Task
- Zend_Scheduler_Task_Rule
- Zend_Scheduler_Backend_Interface
- Zend_Scheduler_Backend_File
- Zend_Scheduler_Backend_Db

8. Use Cases

```
$scheduler = new Zend_Scheduler();

$task    = new Zend_Scheduler_Task();
$request = new Zend_Controller_Request_Http();
$request->setControllerName('thread')
        ->setActionName('prune')
        ->setParam('days' => 90);
$task->setMonths('Jan-Mar, April')
     ->setWeekdays('Friday-Monday')
     ->setHours('20-23')
     ->setMinutes('0/5, 30-33, 59')
     ->setRequest($request);
// 8:00, 8:05, 8:10, 8:15, 8:20, 8:25, 8:30, 8:31, 8:32...
$scheduler->addTask('Prune threads', $task);

$task    = new Zend_Scheduler_Task();
$request = new Zend_Controller_Request_Http();
$request->setControllerName('task')
        ->setActionName('cleanCache');
$task->setMinutes('20-40')
     ->setRequest($request);
$scheduler->addTask('Clean cache', $task);

$scheduler->setController($controller);
$scheduler->run();
```

```
$scheduler = new Zend_Scheduler();
$task      = new Zend_Scheduler_Task();
$request   = new Zend_Controller_Request_Console();
$request->setControllerName('report')
        ->setActionName('create');
$task->setDays('last-1')
     ->setHours('3-5')
     ->setRequest($request);
$scheduler->addTask('Create report', $task);
$scheduler->run($controller);
```

9. Class Skeletons

```


class Zend_Scheduler
{
    /** @var Zend_Date Request time */
    protected $_time = null;

    /** @var array Tasks */
    protected $_tasks = array();

    /** @var array Controller */
    protected $_controller = null;

    /** @var int Maximum allowable tasks to run on a given request */
    protected $_executionLimit = 0;

    /** @var string Backend */
    protected $_backend = null;

    /**
     * Constructor.
     *
```

```
     * @param string $backend Name of backend ('File', 'Db', etc.)
     * @param array $options Backend options
     */
    public function __construct($backend = '', array $options = array());

    /**
     * Set the time (by default, the request time).  For testing purposes a
     * different time can be passed in.
     *
     * @param  Zend_Date $time
     * @return Zend_Scheduler This instance
     */
    public function setTime(Zend_Date $time = null);

    /**
     * Set backend and options.
     *
     * @param string $backend Name of backend ('File', 'Db', etc.)
     * @param array $options Backend options
     */
    public function setBackend($backend, array $options = array());

    /**
     * Sets controller instance.
     *
     * @param  Zend_Controller_Front $controller
     * @return Zend_Scheduler This instance
     */
    public function setController(Zend_Controller_Front $controller);

    /**
     * Adds a task.
     *
     * @param  string $name Task name
     * @param  Zend_Scheduler_Task $task
     * @return Zend_Scheduler This instance
     */
    public function addTask($name, Zend_Scheduler_Task $task);

    /**
     * Checks if scheduler has a task.
     *
     * @param  string $name Task name
     * @return bool True if task by that name has been added
     */
    public function hasTask($name);

    /**
     * Removes a task.
     *
     * @param string $name Task name
     */
    public function removeTask($name);

    /**
     * Sets the maximum allowable tasks to run on a given request.  To allow
     * an infinite number of tasks to run, set to 0.
     *
     * @param  int $limit Task execution limit
     * @return Zend_Scheduler This instance
     */
    public function setExecutionLimit($limit = 0);

    /**
     * Executes all scheduled tasks.
     *
     * @param  Zend_Controller_Front $controller
     * @return array|null Array of Response objects, or null if no tasks
```

```
     */
    public function run(Zend_Controller_Front $controller = null);
}
```

```
class Zend_Scheduler_Task
{
    /** @var Zend_Date Request time */
    protected $_time = null;

    /** @var array Array of Zend_Controller_Request_Abstract objects */
    protected $_requests = array();

    /** @var array Array of Zend_Scheduler_Task_Rule_Abstract objects */
    protected $_rules = array();

    /**
     * Constructor.
     */
    public function __construct();

    /**
     * Set the time (by default, the request time).  For testing purposes a
     * different time can be passed in.
     *
     * @param Zend_Date $time
     * @return Zend_Scheduler_Task This instance
     */
    public function setTime(Zend_Date $time = null);

    /**
     * Sets the request.
     *
     * @param  Zend_Controller_Request_Abstract $request
     * @return Zend_Scheduler_Task This instance
     */
    public function setRequest(Zend_Controller_Request_Abstract $request);

    /**
     * Adds a request.
     *
     * @param  Zend_Controller_Request_Abstract $request
     * @return Zend_Scheduler_Task This instance
     */
    public function addRequest(Zend_Controller_Request_Abstract $request);

    /**
     * Sets the first time the task is scheduled to run.
     *
     * @param  string $time
     * @return Zend_Scheduler_Task This instance
     */
    public function setFirstRun($time);

    /**
     * Sets the months in which the task is scheduled to run.
     *
     * @param  string $months Comma-delimited list of months (e.g., January)
     * @return Zend_Scheduler_Task This instance
     */
    public function setMonths($months);

    /**
     * Sets the days of the month in which the task is scheduled to run.
     *
     * @param  string $days Comma-delimited list of days (1-31)
```

```
     * @return Zend_Scheduler_Task This instance
     */
    public function setDays($days);

    /**
     * Sets the days of the week in which the task is scheduled to run.
     *
     * @param  string $days Comma-delimited list of days of the week (e.g., Monday)
     * @return Zend_Scheduler_Task This instance
     */
    public function setWeekdays($weekdays);

    /**
     * Sets the hours in which the task is scheduled to run.
     *
     * @param  string $days Comma-delimited list of hours (0-23)
     * @return Zend_Scheduler_Task This instance
     */
    public function setHours($hours);

    /**
     * Sets the minutes in which the task is scheduled to run.
     *
     * @param  string $days Comma-delimited list of minutes (0-59)
     * @return Zend_Scheduler_Task This instance
     */
    public function setMinutes($minutes);

    /**
     * Sets the final time the task is scheduled to run.
     *
     * @param  string $time
     * @return Zend_Scheduler_Task This instance
     */
    public function setFinalRun($time);

    /**
     * Determines whether a task should be ran or not.
     *
     * @return bool
     */
    public function isScheduled();

    /**
     * Dispatches all scheduled requests.
     *
     * @param  Zend_Controller_Front $controller
     * @return array|null Array of Response objects, or null if no tasks
     */
    public function run(Zend_Controller_Front $controller);

    /**
     * Adds a rule.  Called by {@link setMonths()}, {@link setDays()}, etc.
     *
     * @param  string $type Rule type
     * @param  string $rule Rule string
     * @return Zend_Scheduler_Task This instance
```

```
     */
    protected function _addRule($type, $rule);
}
```

```
class Zend_Scheduler_Task_Rule
{
    /** @var Zend_Date Request time */
    protected $_time = null;

    /** @var string Type */
    protected $_type = '';

    /** @var string Rule */
    protected $_rule = '';

    /** @var array Time numeric values */
    protected $_times = array(
        'months'   => Zend_Date::MONTH_DIGIT,
        'days'     => Zend_Date::DAY_SHORT,
        'weekdays' => Zend_Date::WEEKDAY_8601,
        'hours'    => Zend_Date::HOUR_SHORT,
        'minutes'  => Zend_Date::MINUTE_SHORT
    );

    /** @var array Time ranges */
    protected $_ranges = array(
        'months'   => array('minimum' => 1, 'maximum' => 12),
        'days'     => array('minimum' => 1, 'maximum' => 31),
        'weekdays' => array('minimum' => 1, 'maximum' => 7),
        'hours'    => array('minimum' => 0, 'maximum' => 23),
        'minutes'  => array('minimum' => 0, 'maximum' => 59)
    );

    /** @var array Months of the year */
    protected $_months = array(
        'Jan' => 1, 'Feb' => 2,  'Mar' => 3,  'Apr' => 4,
        'May' => 5, 'Jun' => 6,  'Jul' => 7,  'Aug' => 8,
        'Sep' => 9, 'Oct' => 10, 'Nov' => 11, 'Dec' => 12
    );

    /** @var array Days of the month */
    protected $_days = array(
        'last' => ''
    );

    /** @var array Days of the week */
    protected $_weekdays = array(
        'Sun' => 1, 'Mon' => 2, 'Tue' => 3, 'Wed' => 4,
        'Thu' => 5, 'Fri' => 6, 'Sat' => 7
    );

    /**
     * Constructor.
     *
     * @param string $type
     * @param string $rule
     */
    public function __construct($type, $rule);

    /**
     * Set the time (by default, the request time).  For testing purposes a
     * different time can be passed in.
     *
     * @param Zend_Date $time
     */
```

```
    public function setTime(Zend_Date $time = null);

    /**
     * Checks if rule matches the supplied timestamp.
     *
     * @param  Zend_Date $time
     * @return bool
     */
    public function matches(Zend_Date $time = null);

    /**
     * Gets int value for specified type (e.g., 'Dec' equals 12)
     *
     * @return int
     */
    protected function _getTime();

    /**
     * Parses the rule.
     *
     * @param  string $rule
     * @return array Expanded values
     */
    protected function _parse();

    /**
     * Gets minimally-acceptable value for type.
     *
     * @return int Minimum
     */
    protected function _getMinimum();

    /**
     * Gets maximally-acceptable value for type.
     *
     * @return int Maximum
     */
    protected function _getMaximum();

    /**
     * Checks if rule value is within the acceptable range.
     *
     * @param  int $value
     * @return bool
```

```
     */
    protected function _inRange($value);
}
```

```
interface Zend_Scheduler_Backend_Interface
{
    /**
     * Constructor.
     *
     * @param array $options Backend options
     */
    public function __construct(array $options = array());

    /**
     * Sets the remaining tasks to perform.
     *
     * @param array Remaining tasks
     */
    public function setRemainingTasks(array $tasks = array());

    /**
     * Gets the remaining tasks to perform.
     *
     * @return array Remaining tasks
     */
    public function getRemainingTasks();
}
```

```
class Zend_Scheduler_Backend_File implements Zend_Scheduler_Backend_Interface
{
    /**
     * Constructor.
     *
     * @param array $options Backend options
     */
    public function __construct(array $options = array());

    /**
     * Sets the filename.
     *
     * @param string $filename Name of file in which to record remaining tasks
     */
    protected function _setFilename($filename);

    /**
     * Sets the remaining tasks to perform.
     *
     * @param array $tasks Remaining tasks
     */
    public function setRemainingTasks(array $tasks = array());

    /**
     * Gets the remaining tasks to perform.
     *
     * @return array Remaining tasks
     */
    public function getRemainingTasks();
}
```

```
class Zend_Scheduler_Exception extends Zend_Exception {}
```

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