Zend_Db_Table_Mptt - Nick Pack & Hector Virgen

<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_Db_Table_Mptt Component Proposal

Proposed Component Name	Zend_Db_Table_Mptt
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Db_Table_Mptt
Proposers	Nick Pack, Hector Virgen
Zend Liaison	TBD
Revision	1.3 - 20 August 2009: Refactored Skeleton, Extra UC's. (wiki revision: 19)

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_Db_Table_Mptt is an extension to Zend_Db_Table to implement the Modified Preorder Tree Traversal algorithm

2. References

- Sitepoint Hierarchical Database Article
- Django MPTT
- CakePHP MPTT Component
- Kohana PHP Framework MPTT
- Original Zend_NestedSet Discussion

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** automatically update the left and right values of each node on update, insert and delete
- This component **will** provide a method to retrieve ancestor nodes (e.g. for building breadcrumbs)
- This component **will** provide a method to build a tree of all nodes in hierachy
- This component **will** provide a method of retrieving child nodes of a parent node

- This component **will** provide a method to convert existing data into MPTT

4. Dependencies on Other Framework Components

- Zend_Db_Table
- Zend_Db_Table_Exception

5. Theory of Operation

This component will retrieve hierarchical data with the minimum amount of overhead

6. Milestones / Tasks

- Milestone 1: Complete Skeleton Class/Proposal
- Milestone 2: Working prototype
- Milestone 3: Working prototype checked into the incubator
- Milestone 4: Unit tests exist, work, and are checked into SVN.
- Milestone 5: Initial documentation exists.

7. Class Index

- Zend_Db_Table_Mptt
- Zend_Db_Table_Mptt_Exception

8. Use Cases

UC-01

```
class MyTree extends Zend_Db_Table_Mptt
{
protected $_name = 'tree_example';
protected $_traversal = array(
'left'          => 'lt',
'right'         => 'rt',
'column'        => 'row_id',
'refColumn'     => 'parent_id'
);
}
```

UC-02

```
// Fetch all of the children for row with the ID of 1
$node = $mytree->find(1)->current();
$descendents = $mytree->fetchAllDescendents($node);
// or:
$descendents = $mytree->fetchAllDescendents(1);
```

UC-03

```
// With optional select object
$select = $mytree->select()->where('published = ?', 1)->limit(5);
$descendends = $mytree->fetchAllDescendents($node, $select);
```

UC-04

```
//Fetching nodes as a tree
$tree = $mytree->fetchTree();

foreach ($tree as $node) {
echo str_repeat(' ', $node->tree_depth * 4) . $node->id . PHP_EOL;
}
```

UC-05

```
// make all direct children root nodes, retaining their subtrees
$table->delete($id, Zend_Db_Table_Mptt::DELETE_MAKE_NULL);
```

UC-06

```
// restrict deleting if child nodes exists
if ($table->delete($id, Zend_Db_Table_Mptt::DELETE_RESTRICT)) {
 // delete successful
} else {
 // delete failed
}
```

UC-07

```
// delete the node and all it's child nodes recursively
$table->delete($id, Zend_Db_Table_Mptt::DELETE_CASCADE);
```

UC-08

```
// attach all nodes direct children to it's parent, making them siblings
$table->delete($id, Zend_Db_Table_Mptt::DELETE_REATTACH);
```

UC-09

```
// attach all nodes direct children to node with ID $newParent
$table->delete($id, Zend_Db_Table_Mptt::DELETE_REATTACH, $newParent);
```

9. Class Skeletons

Zend_Db_Table_Mptt

```
<?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_Db_Table_Mptt
 * @copyright  Copyright (c) 2005-2008 Zend Technologies USA Inc. (http://www.zend.com)
 * @license    http://framework.zend.com/license/new-bsd     New BSD License
 * @version
 *
 */

require_once 'Zend/Db/Table/Abstract.php';

class Zend_Db_Table_Mptt extends Zend_Db_Table_Abstract
{
    /**
     * Traversal tree information
     * Values:
     *  'left'          => column name for left value
     *  'right'         => column name for right value
     *  'column'        => column name for identifying row (primary key assumed)
     *  'refColumn'     => column name for parent id (if not set, will look in reference map for own
table match)
     *  'order'         => order by for rebuilding tree (e.g. "`name` ASC, `age` DESC")
     *
     * @var array $_traversal
     */
    protected $_traversal = array();
    /**
     * Automatically is set to true once traversal info is set and verified
     *
     * @var boolean $_isTraversable
     */
    protected $_isTraversable = false;
 /**
  * Delete mode constants
  */
    const DELETE_MAKE_NULL = '_deleteMakenull';
    const DELETE_RESTRICT = '_deleteRestrict';
```

```
    const DELETE_CASCADE = '_deleteCascade';
    const DELETE_REATTACH = '_deleteReattach';

    /**
     * Construct Zend_Db_Table Object & verify traversal capabilities of table
     *
     */
    public function __construct ($config = array())
    {
        parent::__construct($config);
        $this->_initTraversal();
    }
    /**
     * Returns columns names
     *
     * @return array columns
     */
    public function getColumns ()
    {
       return $this->info(Zend_Db_Table_Abstract::COLS);
    }
    /**
     * Returns the table name and schema separated by a dot for use in sql queries
     *
     * @return string schema.name || name
     */
    public function getName ()
    {
       return $this->_schema ? $this->_schema . '.' . $this->_name : $this->_name;
    }
    /**
     * Override delete method, calls _delete prefixed function dependent on $mode
     * @param integer $id
     * @param const $mode
     * @param mixed $newParent
     */
    public function delete ($id, $mode = self::DELETE_RESTRICT, $newParent = null)
    {}
   /**
     * Override insert method - calls _insertTraversable
     *
     * @param mixed $data
     * @return primary key
     */
    public function insert (array $data)
    {}
    /**
     * Override update method, if $newParent is specified rebuild traversal data
     *
     * @param mixed $data
     * @param mixed $where
     * @param mixed $newParent
     * @return int
     */
    public function update (array $data, $where, $newParent = false)
    {}
    /**
     * Public function to rebuild tree traversal. The recursive function
     * _rebuildTreeTraversal() must be called without arguments.
     *
     * @return $this - Fluent interface
     */
    public function rebuildTreeTraversal ()
    {
        $this->_rebuildTreeTraversal();
        return $this;
    }
    /**
     * Recursively rebuilds the modified preorder tree traversal
     * data based on a parent id column
```

```
     *
     * @param int $parentId
     * @param int $leftValue
     * @return int new right value
     */
    protected function _rebuildTreeTraversal ($parentId = null, $leftValue = 0)
    {}
    /**
     * Calculates left and right values for new row and inserts it.
     * Also adjusts all affected rows to make room for the new row.
     *
     * @param array $data
     * @return int $id
     */
    protected function _insertTraversable($data)
    {}
    /**
     * Fetches all descendents of a given node
     *
     * @param Zend_Db_Table_Row_Abstract|string $row - Row object or value of row id
     * @param Zend_Db_Select $select - optional custom select object
     * @return Zend_Db_Table_Rowset|null
     */
    public function fetchAllDescendents ($row, Zend_Db_Select $select = null)
    {}
    /**
     * Fetches all descendents of a given node and returns them as a tree
     *
     * @param Zend_Db_Table_Row_Abstract|string|int $rows- Row object or value of row id or array of
rows
     * @param Zend_Db_Select $select - optional select object
     * @return Zend_Db_Table_Rowset|null
     */
    public function fetchTree ($row = null, Zend_Db_Select $select = null)
    {}
    /**
     * Fetches all ancestors of a given node
     *
     * @param Zend_Db_Table_Row_Abstract|string $row - Row object or value of row id
     * @param Zend_Db_Select $select - optional custom select object
     * @return Zend_Db_Table_Rowset|null
     */
    public function fetchAllAncestors ($row, Zend_Db_Select $select = null)
    {}
    /**
     * Initialise Traversal, Verify that cols supplied in $_traversal exist and are of the correct
type
     *
     * @return void
     */
    protected function _initTraversal ()
    {}
    /**
     * Verifies that the current table is traversable
     *
     * @throws Zend_Db_Exception - Table is not traversable
     */
    protected function _verifyTraversable ()
    {
        if (! $this->_isTraversable) {
            require_once 'Zend/Db/Table/Mptt/Exception.php';
            throw new Zend_Db_Table_Mptt_Exception("Table {$this->_name} is not traversable");
        }
    }

    /**
     * Make all direct child nodes root nodes, retaining their subtrees
     */
    protected function _deleteMakenull()
    {}
```

```
    /**
     * Prevent node deletion if node has children
     */
    protected function _deleteRestrict()
    {}
    /**
     * Delete a node and all its child nodes recursively
     */
    protected function _deleteCascade()
    {}
    /**
     * Attach all child nodes to node with specified parent id
     */
```

```
    protected function _deleteReattach()
    {}
}
```

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