Zend_Paginator_Adapter_DbTable - 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_Paginator_Adapter_DbTable Component Proposal

Proposed Component Name	Zend_Paginator_Adapter_DbTable
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Paginator_Adapter_DbTable
Proposers	Tom Graham
Zend Liaison	TBD
Revision	1.0 - 8 September 2008: Initial Draft. (wiki revision: 3)

Table of Contents

1. Overview

Zend_Paginator_Adapter_DbTable is an extension of Zend_Paginator_Adapter_DbSelect that allows a Zend_Db_Table_Rowset to be returned as the result of the paginator.

2. References

- Example code and usage on my blog

3. Component Requirements, Constraints, and Acceptance Criteria

Most requirements take the form of "foo will do ...." or "foo will not support ...", although different words and sentence structure might be used. Adding functionality to your proposal is requirements creep (bad), unless listed below. Discuss major changes with your team first, and then open a "feature improvement" issue against this component.

- This component **will** correctly reads a developers mind for intent and generate the right configuration file.

- The generated config file **will not** support XML, but will provide an extension point in the API.
- This component **will** use no more memory than twice the size of all data it contains.
- This component **will** include a factory method.
- This component **will not** allow subclassing. (i.e. when reviewed, we expect to see "final" keyword in code)
- This component **will** only generate data exports strictly complying with RFC 12345.
- This component **will** validate input data against formats supported by ZF component Foo.
- This component **will not** save any data using Zend_Cache or the filesystem. All transient data **will be** saved using Zend_Session.

4. Dependencies on Other Framework Components

- Zend_Paginator
- Zend_Paginator_Adapter_DbSelect
- Zend_Paginator_Adapter_Abstract
- Zend_Exception

5. Theory of Operation

The same as DbSelect only a Zend_Db_Table_Rowset is returned.

6. Milestones / Tasks

Describe some intermediate state of this component in terms of design notes, additional material added to this page, and / code. Note any significant dependencies here, such as, "Milestone #3 can not be completed until feature Foo has been added to ZF component XYZ." Milestones will be required for acceptance of future proposals. They are not hard, and many times you will only need to think of the first three below.

- Milestone 1: design notes will be published here
- Milestone 2: Working prototype checked into the incubator supporting use cases #1, #2, ...
- Milestone 3: Working prototype checked into the incubator supporting use cases #3 and #4.
- Milestone 4: Unit tests exist, work, and are checked into SVN.
- Milestone 5: Initial documentation exists.

If a milestone is already done, begin the description with "[DONE]", like this:

- Milestone #: [DONE] Unit tests ...

7. Class Index

- Zend_Paginator_Adapter_DbTable

8. Use Cases

UC-01

```
$table = new Users();
$select = $table->select()->order('last_name ASC');
$paginator = new Zend_Paginator(Zend_Paginator_Adapter_DbTable($select, $table);
```

9. Class Skeletons

```
class Zend_Paginator_Adapter_DbTable extends Zend_Paginator_Adapter_DbSelect
{
    /**
     * The table to select from
     *
     * @var Zend_Db_Table
     */
    protected $_table;

    /**
     * Constructor.
     *
     * @param Zend_Db_Table_Select $select The select query
     * @param Zend_Db_Table $table The table to select from
     */
    public function __construct(Zend_Db_Table_Select $select, Zend_Db_Table $table)
    {
        $this->_select = $select;
        $this->_table = $table;
    }

    /**
     * Returns a rowset object
     *
     * @param  integer $offset Page offset
     * @param  integer $itemCountPerPage Number of items per page
     * @return Zend_Db_Table_Rowset
     */
    public function getItems($offset, $itemCountPerPage)
    {
        $this->_select->limit($itemCountPerPage, $offset);

        return $this->_table->fetchAll($this->_select);
    }

    /**
     * Returns the total number of rows in the result set.
     *
     * @return integer
     */
    public function count()
    {
        if ($this->_rowCount === null) {
            $expression = new Zend_Db_Expr('COUNT(*) AS ' . self::ROW_COUNT_COLUMN);

            $rowCount = clone $this->_select;
            $rowCount->from($this->_table)
                     ->reset(Zend_Db_Select::COLUMNS)
                     ->reset(Zend_Db_Select::ORDER)
                     ->reset(Zend_Db_Select::LIMIT_OFFSET)
                     ->columns($expression);

            $this->setRowCount($rowCount);
```

```
        }

        return $this->_rowCount;
```

```
        }
    }
```

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