Zend_Session_SaveHandler_DbTable

<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_Session_SaveHandler_DbTable Component Proposal

Proposed Component Name	Zend_Session_SaveHandler_DbTable
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Session_SaveHandler_DbTable
Proposers	Jordan Raub Felix Jendrusch Darby Felton, Zend liaison
Revision	• 1.0 - 4 March 2007: Added proposal draft • 1.1 - 20 July 2007: Removed optional Zend_ACL dependency. • 1.2 - 24 July 2007: Removed optional Zend_Auth dependency and clarified Zend_Db_Table to Zend_Session_SaveHandler_DbTable relationships • 1.3 - 30 July 2007: Added Save_Path and Session name support (both optional). Unit tests written for virtually all configuration modes • 1.4 - 1 Aug 2007: Zend_Session_SaveHandler_Db changed to Zend_Session_SaveHandler_DbTable (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_Session_SaveHandler_DbTable is a database independent adapter to save session information in a database

2. References

- http://framework.zend.com/manual/en/zend.session.html
- http://php.net/session
- http://php.net/manual/en/function.session-set-save-handler.php

3. Component Requirements, Constraints, and Acceptance Criteria

- Zend_Session_SaveHandler_DbTable will save session data in a database independent way

4. Dependencies on Other Framework Components

- Zend_Db_Table
- Zend_Session_SaveHandler_Interface

5. Theory of Operation

Since Zend_Session_SaveHandler_Interface extends Zend_Db_Table, all normal table operations are available. Also when instantiating a new Zend_Session_SaveHandler_Interface($config), the $config array is a combined config for Zend_Db_Table and Zend_Session_SaveHandler_Interface. So the 'name' and 'primary' keys in my below use case refer to Zend_Db_Table config keys.

In this table id is the actual session ID given to the user. This session id depends on session.hash_function and session.hash_bits_per_character for length I use sha1 with 5bits/char so 32 chars. The defaults for php are session.hash_function refers to md5 and session.hash_bits_per_character = 4. see http://php.net/session

Session modification time in the implemented tables is currently a Unix Timestamp.

Session data column contains the serialized $_SESSION array, as is in the regular save handler seen in files.

Zend_Session_SaveHandler_DbTable saves all data from a session into a single DB table.

Dump of a basic MySQL tables for use with Zend_Session_SaveHandler_DbTable:

UC-01

```
CREATE TABLE `Sessions` (
`id` varchar(32) NOT NULL,
`modified` int(11) default NULL,
`lifetime` int(11) default NULL,
`data` text,
PRIMARY KEY (`id`)
) ENGINE=InnoDB
```

UC-02

```
CREATE TABLE `Sessions` (
`id` varchar(32) NOT NULL,
`save_path` varchar(32) NOT NULL,
`name` varchar(32) NOT NULL default '',
`modified` int(11) default NULL,
`lifetime` int(11) default NULL,
`data` text,
PRIMARY KEY (`id`,`save_path`,`name`)
) ENGINE=InnoDB;
```

6. Milestones / Tasks

- Milestone 1: [DONE] Write Proposal
- Milestone 2: [DONE] Basic Functionallity working with Unit tests
- Milestone 3: Working prototype checked into the incubator supporting use case #1 along with Unit tests

7. Class Index

- Zend_Session_SaveHandler_DbTable
- Zend_Session_SaveHandler_Exception

8. Use Cases

UC-01

The simplest way to use Zend_Session_SaveHandler_DbTable

```
<?php
//This is in your bootstrap, note you can also use Zend_Config in the constructor
$config = array(
'name'           => 'session',      //table name as per Zend_Db_Table
'primary'        => 'Session_ID',   //the sessionID given by php
'modifiedColumn' => 'modified',     //time the session should expire
'dataColumn'     => 'Session_data', //serialized data
'lifetimeColumn' => 'lifetime'      //end of life for a specific record
);
Zend_Session::setSaveHandler(new Zend_Session_SaveHandler_DbTable($config));
Zend_Session::start();
?>
```

Now you can use Zend_Session as normal, also if you have some legacy code you can use $_SESSION as you normally would. Everything will be saved in the DB table.

UC-02

You can also have a column for the session Name and/or session.save_path

```
<?php
//This is in your bootstrap, note you can also use Zend_Config in the constructor
$config = array(
'name'              => 'session',          //table name as per Zend_Db_Table
'primary'           => array(
                           'Session_ID',       //the sessionID given by php
                           'save_path',        //session.save_path
                           'name',             //session name
                       ),
'primaryAssignment' => array(                   //you must tell the save handler which columns you are
using as the primary key. ORDER IS IMPORTANT
                           'sessionId',        //first column of the primary key is of the sessionID
                           'sessionSavePath',  //second column of the primary key is the save path
                           'sessionName',      //third column of the primary key is the session name
                       ),
'modifiedColumn'    => 'modified',             //time the session should expire
'dataColumn'        => 'Session_data',         //serialized data
'lifetimeColumn'    => 'lifetime',             //end of life for a specific record
);
Zend_Session::setSaveHandler(new Zend_Session_SaveHandler_DbTable($config));
Zend_Session::start();
?>
```

UC-03

You can also make sessions persistant

```
<?php
//This is in your bootstrap, note you can also use Zend_Config in the constructor
$config = array(
'name'           => 'session',      //table name as per Zend_Db_Table
'primary'        => 'Session_ID',   //the sessionID given by php
'modifiedColumn' => 'modified',     //time the session should expire
'dataColumn'     => 'Session_data', //serialized data
'lifetimeColumn' => 'lifetime'      //end of life for a specific record
);
$saveHandler = new Zend_Session_SaveHandler_DbTable($config);
//cookie persist for 30 days
Zend_Session::rememberMe($seconds = (60 * 60 * 24 * 30));

//make the session persist for 30 days
$savehandler->setLifetime($seconds)
    ->setOverrideLifetime(true);
//similarly,
//$saveHandler->setLifetime($seconds, true);
Zend_Session::setSaveHandler($saveHandler);
Zend_Session::start();
?>
```

9. Class Skeletons

```
class Zend_Session_SaveHandler_Exception extends Zend_Session_Exception{}
```

```
<?php
/**
 * Zend_Session_SaveHandler_DbTable
 *
```

```
 * @category   Zend
 * @package    Zend_Session
 * @copyright  Copyright (c) 2007 Jordan Raub <ludicruz@yahoo.com> and Felix Jendrusch
<felixjendrusch@googlemail.com>
 * @license    http://framework.zend.com/license/new-bsd     New BSD License
 */
class Zend_Session_SaveHandler_DbTable extends Zend_Db_Table_Abstract implements
Zend_Session_SaveHandler_Interface
{
    /**
     * see __construct() comments
     */
    const PRIMARY_ASSIGNMENT                   = 'primaryAssignment';
    const PRIMARY_ASSIGNMENT_SESSION_SAVE_PATH = 'sessionSavePath';
    const PRIMARY_ASSIGNMENT_SESSION_NAME      = 'sessionName';
    const PRIMARY_ASSIGNMENT_SESSION_ID        = 'sessionId';

    const MODIFIED_COLUMN   = 'modifiedColumn';
    const LIFETIME_COLUMN   = 'lifetimeColumn';
    const DATA_COLUMN       = 'dataColumn';

    const LIFETIME          = 'lifetime';
    const OVERRIDE_LIFETIME = 'overrideLifetime';

    const PRIMARY_TYPE_NUM         = 'PRIMARY_TYPE_NUM';
    const PRIMARY_TYPE_PRIMARYNUM  = 'PRIMARY_TYPE_PRIMARYNUM';
    const PRIMARY_TYPE_ASSOC       = 'PRIMARY_TYPE_ASSOC';
    const PRIMARY_TYPE_WHERECLAUSE = 'PRIMARY_TYPE_WHERECLAUSE';

    /**
     * Session table primary key value assignment
     *
     * @var array
     */
    protected $_primaryAssignment = null;

    /**
     * Session table last modification time column
     *
     * @var string
     */
    protected $_modifiedColumn = null;

    /**
     * Session table lifetime column
     *
     * @var string
     */
    protected $_lifetimeColumn = null;

    /**
     * Session table data column
     *
     * @var string
     */
    protected $_dataColumn = null;

    /**
     * Session lifetime
     *
     * @var int
     */
    protected $_lifetime = false;

    /**
     * Whether or not the lifetime of an existing session should be overridden
     *
     * @var boolean
     */
    protected $_overrideLifetime = false;
```

```
    /**
     * Session save path
     *
     * @var string
     */
    protected $_sessionSavePath;

    /**
     * Session name
     *
     * @var string
     */
    protected $_sessionName;

    /**
     * Constructor
     *
     * $config is an instance of Zend_Config or array of key/value pairs containing configuration
options for Zend_Session_SaveHandler_DbTable
     * and Zend_Db_Table_Abstract. These are the configuration options for
Zend_Session_SaveHandler_DbTable:
     *
     * primaryAssignment => (string|array) Session table primary key value assignment (optional;
default: 1 => sessionId)
     *                          You have to assign a value to each primary key of your session table.
The value of this configuration
     *                          option is either a string if you have only one primary key or an array
if you have multiple primary keys.
     *                          The array consists of numeric keys starting at 1 and string values.
There are some values which will be
     *                          replaced by session information:
     *
     *                              sessionId       => The id of the current session
     *                              sessionName     => The name of the current session
     *                              sessionSavePath => The save path of the current session
     *
     *                          NOTE: One of your assignments MUST contain 'sessionId' as value!
     *
     * modifiedColumn    => (string) Session table last modification time column
     * lifetimeColumn    => (string) Session table lifetime column
     * dataColumn        => (string) Session table data column
     * lifetime          => (integer) Session lifetime (optional; default:
ini_get('session.gc_maxlifetime'))
     * overrideLifetime  => (boolean) Whether or not the lifetime of an existing session should be
overridden (optional; default: false)
     *
     * @param Zend_Config|array $config      User-provided configuration
     * @return void
     * @throws Zend_Session_SaveHandler_Exception
     */
    public function __construct($config) {}

    /**
     * Destructor
     *
     * @return void
     */
    public function __destruct()
    {
        Zend_Session::writeClose();
    }

    /**
     * Set session lifetime and optional whether or not the lifetime of an existing session should be
overridden
     *
     * $lifetime === false resets lifetime to session.gc_maxlifetime
     *
     * @param int $lifetime
```

```
     * @param boolean $overrideLifetime (optional)
     * @return Zend_Session_SaveHandler_DbTable     */
    public function setLifetime($lifetime, $overrideLifetime = null) {}

    /**
     * Retrieve session lifetime
     *
     * @return int
     */
    public function getLifetime() {}

    /**
     * Set whether or not the lifetime of an existing session should be overridden
     *
     * @param boolean $overrideLifetime
     * @return Zend_Session_SaveHandler_DbTable
     */
    public function setOverrideLifetime($overrideLifetime) {}

    /**
     * Retrieve whether or not the lifetime of an existing session should be overridden
     *
     * @return boolean
     */
    public function getOverrideLifetime() {}

    /**
     * Open Session
     *
     * @param string $save_path
     * @param string $name
     * @return boolean
     */
    public function open($save_path, $name) {}

    /**
     * Close session
     *
     * @return boolean
     */
    public function close() {}

    /**
     * Read session data
     *
     * @param string $id
     * @return string
     */
    public function read($id) {}

    /**
     * Write session data
     *
     * @param string $id
     * @param string $data
     * @return boolean
     */
    public function write($id, $data) {}

    /**
     * Destroy session
     *
     * @param string $id
     * @return boolean
     */
    public function destroy($id) {}

    /**
     * Garbage Collection
     *
```

```
 * @param int $maxlifetime
 * @return true
 */
```

```
    public function gc($maxlifetime) {}
  }
  ?>
```

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