Zend_Auth_Adapter_Doctrine_Table - Jason Eisenmenger & David A. Werner

<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_Auth_Adapter_Doctrine_Table Component Proposal

Proposed Component Name	Zend_Auth_Adapter_Doctrine_Table
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Auth_Adapter_Doctrine_Table
Proposers	Jason Eisenmenger David A. Werner
Zend Liaison	Ralph Schindler
Revision	1.0 - 16 June 2008: Working Draft. (wiki revision: 10)

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_Auth_Adapter_Doctrine_Table is an authentication adapter for Zend_Auth to use the Doctrine DBAL

You may download the most up-to-date class here.

2. References

- [Zend_Auth_Adapter_Dbtable]

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** provide the exact same functionality as Zend_Auth_Adapter_DbTable

- This component **will** support Doctrine's query conventions in the adapter methods.

4. Dependencies on Other Framework Components

- Zend_Auth_Adapter_Interface
- Zend_Auth_Result
- Zend_Auth_Adapter_Exception
- Doctrine_Connection_Common
- Doctrine_Query

5. Theory of Operation

This component is used exactly the same way as Zend_Auth_Adapter_DbTable, with the added functionality of using Doctrine's preferred way of casting names.

6. Milestones / Tasks

- Milestone 1: [DONE] Initial proposal published for review with working prototype. (See overview for link)
- 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.

7. Class Index

- Zend_Auth_Adapter_Doctrine_Table

8. Use Cases

UC-01

```
// Standard drop-in from Zend_DbTable

        $authAdapter = new Zend_Auth_Adapter_Doctrine_Table(
            Doctrine::getConnectionByTableName('User'));
        $authAdapter->setTableName('User')
                    ->setIdentityColumn('user_email')
                    ->setCredentialColumn('user_password')
                    ->setCredentialTreatment('md5(?)')
                    ->setIdentity('user')
                    ->setCredential('password');
        $auth = Zend_Auth::getInstance();
        $result = $auth->authenticate($authAdapter);
        if ($result->isValid()) {
            $data = $authAdapter->getResultRowObject('user_id', 'user_password');
            $auth->getStorage()->write($data);
        } else {
            echo "No matching email/password";
        }
}
```

UC-02

```
        $authAdapter = new Zend_Auth_Adapter_Doctrine_Table(
            Doctrine::getConnectionByTableName('User'));
        $authAdapter->setTableName('User u')
                    ->setIdentityColumn('u.user_email')
                    ->setCredentialColumn('u.user_password')
                    ->setCredentialTreatment('md5(?)')
                    ->setIdentity('user')
                    ->setCredential('password');
        $auth = Zend_Auth::getInstance();
        $result = $auth->authenticate($authAdapter);
        if ($result->isValid()) {
            $data = $authAdapter->getResultRowObject('user_id', 'user_password');
            $auth->getStorage()->write($data);
        } else {
            echo "No matching email/password";
        }
```

9. Class Skeletons

```
class Zend_Auth_Adapter_Doctrine_Table implements Zend_Auth_Adapter_Interface
{
    /**
     * Database Connection
     *
```

```
     * @var Zend_Db_Adapter_Abstract
     */
    protected $_conn = null;

    /**
     * $_tableName - the table name to check
     *
     * @var string
     */
    protected $_tableName = null;

    /**
     * $_identityColumn - the column to use as the identity
     *
     * @var string
     */
    protected $_identityColumn = null;

    /**
     * $_credentialColumns - columns to be used as the credentials
     *
     * @var string
     */
    protected $_credentialColumn = null;

    /**
     * $_identity - Identity value
     *
     * @var string
     */
    protected $_identity = null;

    /**
     * $_credential - Credential values
     *
     * @var string
     */
    protected $_credential = null;

    /**
     * $_credentialTreatment - Treatment applied to the credential, such as MD5() or
PASSWORD()
     *
     * @var string
     */
    protected $_credentialTreatment = null;

    /**
     * $_authenticateResultInfo
     *
     * @var array
     */
    protected $_authenticateResultInfo = null;

    /**
     * $_resultRow - Results of database authentication query
     *
     * @var array
     */
```

```
    protected $_resultRow = null;

    /**
     * __construct() - Sets configuration options
     *
     * @param  Zend_Db_Adapter_Abstract $zendDb
     * @param  string                   $tableName
     * @param  string                   $identityColumn
     * @param  string                   $credentialColumn
     * @param  string                   $credentialTreatment
     * @return void
     */
    public function __construct(Doctrine_Connection_Common $conn = null, $tableName =
null, $identityColumn = null,
                                $credentialColumn = null, $credentialTreatment = null)
    {
    }

    /**
     * setConnection() - set the connection to the database
     *
     * @param  Doctrine_Connection_Common $conn
     * @return Zend_Auth_Adapter_Doctrine_Table Provides a fluent interface
     */
    public function setConnection(Doctrine_Connection_Common $conn)
    {
    }

    /**
     * getConnection() - get the connection to the database
     *
     * @return Doctrine_Connection_Common|null
     */
    public function getConnection()
    {
    }

    /**
     * setTableName() - set the table name to be used in the select query
     *
     * @param  string $tableName
     * @return Zend_Auth_Adapter_Doctrine_Table Provides a fluent interface
     */
    public function setTableName($tableName)
    {
    }

    /**
     * setIdentityColumn() - set the column name to be used as the identity column
     *
     * @param  string $identityColumn
     * @return Zend_Auth_Adapter_Doctrine_Table Provides a fluent interface
     */
    public function setIdentityColumn($identityColumn)
    {
    }

    /**
     * setCredentialColumn() - set the column name to be used as the credential column
```

```
     *
     * @param  string $credentialColumn
     * @return Zend_Auth_Adapter_Doctrine_Table Provides a fluent interface
     */
    public function setCredentialColumn($credentialColumn)
    {
    }

    /**
     * setCredentialTreatment() - allows the developer to pass a parameterized string
that is
     * used to transform or treat the input credential data
     *
     * In many cases, passwords and other sensitive data are encrypted, hashed,
encoded,
     * obscured, or otherwise treated through some function or algorithm. By
specifying a
     * parameterized treatment string with this method, a developer may apply
arbitrary SQL
     * upon input credential data.
     *
     * Examples:
     *
     *  'PASSWORD(?)'
     *  'MD5(?)'
     *
     * @param  string $treatment
     * @return Zend_Auth_Adapter_Doctrine_Table Provides a fluent interface
     */
    public function setCredentialTreatment($treatment)
    {
    }

    /**
     * setIdentity() - set the value to be used as the identity
     *
     * @param  string $value
     * @return Zend_Auth_Adapter_Doctrine_Table Provides a fluent interface
     */
    public function setIdentity($value)
    {
    }

    /**
     * setCredential() - set the credential value to be used, optionally can specify a
treatment
     * to be used, should be supplied in parameterized form, such as 'MD5(?)' or
'PASSWORD(?)'
     *
     * @param  string $credential
     * @return Zend_Auth_Adapter_Doctrine_Table Provides a fluent interface
     */
    public function setCredential($credential)
    {
    }

    /**
     * getResultRowObject() - Returns the result row as a stdClass object
     *
```

```
     * @param  string|array $returnColumns
     * @param  string|array $omitColumns
     * @return stdClass|boolean
     */
    public function getResultRowObject($returnColumns = null, $omitColumns = null)
    {
    }

    /**
     * authenticate() - defined by Zend_Auth_Adapter_Interface.  This method is called
to
     * attempt an authentication.  Previous to this call, this adapter would have
already
     * been configured with all necessary information to successfully connect to a
database
     * table and attempt to find a record matching the provided identity.
     *
     * @throws Zend_Auth_Adapter_Exception if answering the authentication query is
impossible
     * @return Zend_Auth_Result
     */
    public function authenticate()
    {
    }

    /**
     * _authenticateSetup() - This method abstracts the steps involved with making
sure
     * that this adapter was indeed setup properly with all required peices of
information.
     *
     * @throws Zend_Auth_Adapter_Exception - in the event that setup was not done
properly
     * @return true
     */
    protected function _authenticateSetup()
    {
    }

    /**
     * _authenticateCreateSelect() - This method creates a Zend_Db_Select object that
     * is completely configured to be queried against the database.
     *
     * @return Doctrine_Query
     */
    protected function _authenticateCreateSelect()
    {
    }

    /**
     * _authenticateQuerySelect() - This method accepts a Doctrine_Query object and
     * performs a query against the database with that object.
     *
     * @param Doctrine_Query $dbSelect
     * @throws Zend_Auth_Adapter_Exception - when a invalid select object is
encoutered
     * @return array
     */
    protected function _authenticateQuerySelect(Doctrine_Query $dbSelect)
```

```
    {
    }

    /**
     * _authenticateValidateResultSet() - This method attempts to make certian that
only one
     * record was returned in the result set
     *
     * @param array $resultIdentities
     * @return true|Zend_Auth_Result
     */
    protected function _authenticateValidateResultSet(array $resultIdentities)
    {
    }

    /**
     * _authenticateValidateResult() - This method attempts to validate that the
record in the
     * result set is indeed a record that matched the identity provided to this
adapter.
     *
     * @param array $resultIdentity
     * @return Zend_Auth_Result
     */
    protected function _authenticateValidateResult($resultIdentity)
    {
    }

    /**
     * _authenticateCreateAuthResult() - This method creates a Zend_Auth_Result object
     * from the information that has been collected during the authenticate() attempt.
     *
     * @return Zend_Auth_Result
     */
    protected function _authenticateCreateAuthResult()
    {
    }
```

```
}
```

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