Zend_Db_Document

<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_Document Component Proposal

Proposed Component Name	Zend_Db_Document
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Db_Document
Proposers	Rostislav Mykhajliw
Zend Liaison	http://github.com/necromant2005/zeus
Revision	1.2 - 31 December 2010: Beta. (wiki revision: 41)

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_Document implements abstract interface to nosql storages.

2. References

- MongoDB
- CouchDB
- Riak
- ProjectVoldemort

- NoSQL patterns
- NoSQL application development
- Google Map-Reduce

3. Component Requirements, Constraints, and Acceptance Criteria

Currently Zend haven't got any adapters to NOSQL storages, but a lot of real application is needed this. Cause NOSQL stoges is more flexible

than SQL RDMS, and provide to build a most powerful and performance application.

- This component **will** provide adapters to NOSQL stores
- This component **will** provide get/set operation
- This component **will** provide update/save document operation
- This component **will** provide MapRecude query interface
- This component **will** provide Document object (similar to Zend_Db_Table_Row_Abstract)

- This component **will** provide transformation adapters to support Zend_Db_Table

4. Dependencies on Other Framework Components

- Zend_Json
- Zend_Http_Client
- Zend_Protobuff

5. Theory of Operation

The primary goal of this component is provide abstract interface to NoSQL storages, witch provide functional by get/set, search, map-reduce queries and very easy interface to update/save/validate document to storage. To do this, we must take several factors into account:

- Transform output result set to array
- Query languages are usually JavaScript, and How exactly build queries the most interesting is map-reduce queries. Cause for simple queries we can develop interface similar to Zend_Db_Select , witch current Adapter will transform into himself query language.
- It must provide full interface to direct queries
- In the first step adapter will provide simple interface get/set by key and direct interface for query.
- Version of documents (in general storages return system field _rev for revision and _id for UID of document)

- Integration with exist Zend\Db\Table
- Integration validation document structure

6. Milestones / Tasks

The first implement most popular nosql storages MongoDB/CouchDB and later all exists.

- Milestone 1: [DONE] create abstract classes storages and document
- Milestone 2: [DONE] build base functionality get/set by key
- Milestone 3: [DONE] integrate query builder
- Milestone 4: integrate query builder for map reduce
- Milestone 5: do new adapters and go to Milestone 1 .

7. Class Index

- Zend\Db\Document (factory class)
- Zend\Db\Document\AbtractAdapter
- Zend\Db\Document\AbstractDocument
- Zend\Db\Document\AbstractCollection

- Zend\Db\Document\Adapter\Mongo
- Zend\Db\Document\Adapter\Couch
- Zend\Db\Document\Adapter\Riak
- Zend\Db\Document\Adapter\Voldemort

- Zend\Db\Document\Document\Mongo

- Zend\Db\Document\Document\Couch
- Zend\Db\Document\Document\Riak
- Zend\Db\Document\Document\Voldemort

- Zend\Db\Document\Collection\Mongo
- Zend\Db\Document\Collection\Couch
- Zend\Db\Document\Collection\Riak
- Zend\Db\Document\Collection\Voldemort

8. Use Cases

UC-01

Factory

```
$adapter = Zend\Db\Document::factory('MongoDb', array(
    'host'   => '127.0.0.1',
    'dbname' => 'dbtest',
));
```

UC-02

Use adapter for get object and store

```
// as array It's not very beautiful cause for revistion you must store technical information "_rev",
"_id" for document
$adapter = Zend\Db\Document::factory('Couch', array( ... ));
$user = $adapater->getCollection('test')->get('peter');
$user['age'] = 24;
$adapter->save($user);

// OO, user know all about itself.
$adapter = Zend_Db_Document::factory('Couch', array( ... ));
$user = $adapater->getCollection('test')->get('peter');
$user->age = 24;
// or ArrayAccess interface
$user['age'] = 24;
$adapter->save($user);


// OO
$adapter = Zend_Db_Document::factory('CouchDb', array( ... ));
$user = $adapater->getCollection('test')->get('peter');
$user->age = 24;
$user->save();
```

UC-07

Create new document with document id

```
$adapter->put('user1234', array(
   'name' => 'bob',
   'age'  => 37,
));
```

UC-08

Create new document with auto generated document id

```
$adapter->put(array(
   'name' => 'bob',
   'age'  => 37,
));
```

9. Class Skeletons

Exceptions

```
class Zend\Db\Document\Exception extends Zend\Exception {}

class Zend\Db\Document\Adapter\Exception extends Zend\Db\Document\Exception {}
```

```
class Zend\Db\Document
{
    public static function factory($adapterName, $adapterOptions)
    {}
}
```

Zend\Db\Document\AbstractAdapter

```
abstract class Zend\Db\Document\AbstractAdapter
{
    protected $_connection = null;

    public function __construct()
    {}

    public function connect()
    {}

    public function disconnect()
    {}

    protected abstract function _connect()
    {}

    protected abstract function _disconnect()
    {}

    public function delete($collectionName, $name)
    {}

    public function get($collectionName, $name)
    {}

    public function put($collectionName, $name, $data)
    {}

    public function post($collectionName, $data)
    {}

    public function findOne($collectionName)
    {}

    public function find($collectionName)
    {}
}
```

Zend\Db\Document\AsbtractDocument

```
class Zend\Db\Document\AbstractDocument
{
    public function __construct($data)
    {}

    public function setCollection()
    {}

    public function getCollection()
    {}

    public function save()
    {}

    public function setFromArray()
    {}

    public function toArray()
    {}

    public function getId()
    {}

    public function __get($name)
    {}

    public function __set($name, $value)
    {}

    public function __isset($name)
    {}
}
```

Zend\Db\Document\AbstractCollection

```
class Zend\Db\Document\AbstractCollection
{
    public function __construct($adapter, $name)
    {}

    public function setAdapter()
    {}

    public function getAdapter()
    {}

    public function get($name)
    {}

    public function put($name, $data)
    {}

    public function post($data)
    {}

    public function delete($name)
    {}

    public function save($document)
    {}
}
```

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