Zend_Nosql_Mongo - Valentin Golev

<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_Nosql_Mongo Component Proposal

Proposed Component Name	Zend_Nosql_Mongo
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Nosql_Mongo
Proposers	Valentin Golev
Zend Liaison	TBD
Revision	1.0 - 6 January 2010: Initial 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_Nosql_Mongo is a namespace containing wrappers and adapters for PHP Mongo classes, intended to make them more handy, simple and compatible with other ZF classes, such as Zend_Paginator.

2. References

- MongoDB official site
- Mongodloid, a library developed by the author of the proposal

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** expose the entire MongoDB API
- This component **will** expose even the API not supported by PHP driver if it is exposed by Mongo DB commands
- This component **will** provide class for a Mongo Object.
    - In a mongo object **will** be only string, int, bool, double, null, array (of any of these types, recursively), Mongo Object, date, MongoID, binary, regexp and JS Code fields. See also Data Types in Mongo
    - Fields **will** be able to be accessed as a member variables, like $object->subObject->subObjField for { subObject : { subObjField :

17 } }.
- This component **will** provide objects for individual documents as well as document collections
  - Document objects **will** allow arbitrary fields
  - Document objects **will** provide accessors for document IDs
  - Document objects **will** allow arbitrary fields
  - Document objects **will** be instances of a class which extends BSON Object
  - Document objects **will** allow atomic operations like increment
  - Document Collections **will** be iterable
  - Document Collections **will** be countable
  - Document Collections **will** allow adding and removing documents
  - Document Collections **will** allow retrieving documents by ID
  - Document Collections **will** allow prohibition of arbitrary fields
  - Document Collections **will** allow strong client-side typing of fields, like 'int' or 'int|array(int)|null'
  - Document Collections **will** allow validators for fields
- This component **will** allow using GridFS for storing files
  - This component **will** be able to be used with Zend MVC
- This component **will** provide a special object for Query
  - Query **will** be countable
  - Query **will** be iterable (by returning Cursor for iteration)
- This component **will** provide a special object for Cursor, which is returned by executing Query
  - Cursor **will** be countable
  - Cursor **will** be iterable
- This component **will** allow to set a specific class for documents in specific collections, as soon as it implements a special interface
  - Cursor and Collection **will** return instances of these specific classes instead of usual Document classes

4. Dependencies on Other Framework Components

- Zend_Exception
- Zend_Validate
- Zend_Paginator
- Zend_Date
- Zend_Controller (for using GridFS with MVC)

5. Theory of Operation

Users create a link to Database, providing connection and db information to the Zend_Nosql_Mongo_Db.
Then they can get a link to a specific collection (which will be created, if not exists).

Users can create a Zend_Nosql_Mongo_Object from any array, as soon as all its values (recursively) can be converted to Mongo types. They can save Zend_Nosql_Mongo_Object to any collection. Saving operation, as well as retrieving, returns an instance of Zend_Nosql_Mongo_Document, which has a specific _id and knows a name of its collection.

Users can query any collection. They can add more and more conditions to the query. When they are done, they can transform the query to a cursor and iterate or array-access it.

Users can set a specific class for a document of specific collection. It must implement Zend_Nosql_Mongo_Document_Interface or, even better, inherit from Zend_Nosql_Mongo_Document. In this case, all cursors from queries on this collection will return the specified class.

Users can make document schema more strict by providing validators, strong typing and disallowing arbitrary fields for any collection. Values of each typed field will be tested if it has one of specified types. If not, it will be converted to the first of the type in the list. If the conversion fails and 'null' is not allowed for the field, the exception is thrown instead of saving a document.

6. Milestones / Tasks

- Milestone 1: Initial proposal and prototype
- Milestone 2: All unit tests

- Milestone 3: Working code, all unit tests passed
- Milestone 4: Documentation

7. Class Index

- Zend_Nosql_Mongo_Db
- Zend_Nosql_Mongo_Collection
- Zend_Nosql_Mongo_Document
- Zend_Nosql_Mongo_Document_Interface
- Zend_Nosql_Mongo_Object
- Zend_Nosql_Mongo_Object_Interface

- Zend_Nosql_Mongo_Query
- Zend_Nosql_Mongo_Cursor

- Zend_Nosql_Mongo_Id
- Zend_Nosql_Mongo_Date
- Zend_Nosql_Mongo_Regexp
- Zend_Nosql_Mongo_Code

- Zend_Nosql_Mongo_Gridfs
- Zend_Nosql_Mongo_Gridfs_File
- Zend_Nosql_Mongo_Gridfs_Cursor

- Zend_Nosql_Mongo_Gridfs_Mvc?

8. Use Cases

```
$db = Zend_Nosql_Mongo_Db(null, 'testdb');
$coll = $db->getCollection('testcollection');
for ($i = 0; $i < 100; $i++) {
    echo $coll->save(new Zend_Nosql_Mongo_Object(array('field' => $i))->getId(), " is another unique
id\n";
}

$q = $coll->ensureIndex('field')->query()->lowerThan('field', 10);
foreach ($q as $item) echo $item->field, " must be lower than 10!\n";
```

9. Class Skeletons

```
class Zend_Nosql_Mongo_Db {
    public function __construct($host = null, $dbname = null) {}
    public function setOptions($options) {}
    public function getCollection($collectionName) {}
}
class Zend_Nosql_Mongo_Collection
 implements IteratorAggregate, Countable {
    public function __construct($db = null, $name = null) {}
    public function setOptions($options) {}
    public function setDocumentClass() {}

    public function ensureIndex($field, $direction) {}

    public function save(Zend_Nosql_Mongo_Object_Interface $object) {}

    public function query() {}
```

```
    public function count() {}
    public function getIterator() {}

    public function setFieldTypes($field, $types) {}
    public function setFieldsTypes($fieldstypes) {}
    public function setFieldValidator($field, $validator) {}
}
class Zend_Nosql_Mongo_Document
 extends Zend_Nosql_Mongo_Object
 implements Zend_Nosql_Mongo_Document_Interface {

 public function __construct($data, Zend_Nosql_Mongo_Collection $collection) {}

 public function getId() {}
 public function getCollection() {}

 public function save() {}

 public function isChanged() {}
 public function commit() {}

 /*
  * and functions for atomic operations
  * see http://www.mongodb.org/display/DOCS/Updating#Updating-ModifierOperations
  * see http://code.google.com/p/mongodloid/
  */
}
interface Zend_Nosql_Mongo_Document_Interface
 extends Zend_Nosql_Mongo_Object_Interface {
 public function __construct($data, Zend_Nosql_Mongo_Collection $collection) {}

 public function getId() {}
 public function getCollection() {}

 public function save() {}

 public function isChanged() {}
 public function commit() {}

 //and functions for atomic operations
}
class Zend_Nosql_Mongo_Object {
 public function __construct($data) {}
 function __get() {}
 function __set() {}
}
interface Zend_Nosql_Mongo_Object_Interface {
 public function __construct($data) {}
}

class Zend_Nosql_Mongo_Query
 implements IteratorAggregate, Countable {
 public function __construct(Zend_Nosql_Mongo_Collection $collection) {}

 public function getCursor() {}

 public function count() {}
    public function getIterator() {}

    /*
     * and functions for all conditions
     * see http://www.mongodb.org/display/DOCS/Advanced+Queries
     * see http://code.google.com/p/mongodloid/
     */
}

class Zend_Nosql_Mongo_Cursor
 implements Iterator, Countable {
 public function __construct(Zend_Nosql_Mongo_Collection $collection, $query) {}
```

```
public function current() {}
public function key() {}
public function next() {}
public function rewind() {}
```

```
  public function valid() {}
  public function count() {}
 }
```

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