Zend_Couch - Matthew Weier O'Phinney

<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_Couch Component Proposal

Proposed Component Name	Zend_Couch
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Couch
Proposers	Matthew Weier O'Phinney
Zend Liaison	TBD
Revision	1.0 - 30 September 2008: Initial Draft. (wiki revision: 3)

Table of Contents

1. Overview

Zend_Couch is intended to be a wrapper around the CouchDB REST API. It will provide methods for all API methods, utilizing Zend_Http_Client for communication with the CouchDB server, and Zend_Json for parsing and creating payloads.

2. References

- CouchDB site
- Phly_Couch implementation

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **MUST** expose the entire CouchDB API, including:
  - Server metadata queries
  - Database manipulation: querying metadata, adding and dropping databases
  - Document manipulation: saving, retrieving, and deleting documents, including bulk updates/inserts
  - View manipulation: creating and querying views, including ad hoc views
- This component **MUST** provide objects for individual documents as well as document sets

        - Document objects **MUST** allow arbitrary fields
        - Document objects **MUST** provide accessors for document IDs and revisions
        - Document Sets **MUST** be iterable
        - Document Sets **MUST** be countable
        - Document Sets **MUST** allow adding and removing documents
        - Document Sets **MUST** allow retrieving documents by ID
- This component **MUST** provide a Result object for API methods that do not return documents or document sets
    - The Result object **MUST** provide access to the response
    - The Result object **MUST** provide access to any JSON data returned in the response
        - The Result object **SHOULD** provide access to any JSON data returned in the response via overloading

4. Dependencies on Other Framework Components

- Zend_Exception
- Zend_Http_Client
- Zend_Json

5. Theory of Operation

Users will instantiate Zend_Couch objects, providing connection information, but using sane defaults if none is provided. A default HTTP client may be set statically, attached individually to Zend_Couch instances, or an HTTP client will be instantiated at first request. Connection information may also be provided after instantiation by either calling individual mutators or using the setOptions() or setConfig() methods.

Once the object is instantiated, the user may start communicating with the CouchDB server via API methods. The Server and Database API methods either do not require arguments, or require simple string arguments; in all such cases, these methods will throw an exception on failure, or return a Result object on success. The user may then query the result object for details of the transaction.

To add a document to the database, the developer may either create an associative array of fields representing the document, or, more preferable, use Zend_Couch_Document to do so. The latter will have methods for specifying the document ID and revision, if available, and use overloading to allow specifying arbitrary document fields. The document is then passed to the docSave() method to save it to the database. When using the docBulkSave() method, the developer would attach one or more documents to a Zend_Couch_DocumentSet – or group them in an array.

When retrieving a single document, a Zend_Couch_Document will be returned.

When retrieving all documents or a subset of documents via a view or query, a Zend_Couch_DocumentSet will be returned. The developer may then iterate over the set.

6. Milestones / Tasks

- Milestone 1: Initial prooposal and prototype code
- Milestone 2: Working code with all tests
- Milestone 3: Documentation

7. Class Index

- Zend_Couch
- Zend_Couch_Document
- Zend_Couch_DocumentSet
- Zend_Couch_Exception
- Zend_Couch_Result

8. Use Cases

UC-01

```
$couch = new Zend_Couch();

// Get server information
$result = $couch->serverInfo();
echo $result->version;

// Get list of all databases
$result = $couch->allDbs();
var_export($result->toArray());
```

UC-02

```
$couch = new Zend_Couch();

try {
    $result = $couch->dbCreate('wiki');
} catch (Zend_Couch_Exception $e) {
    die('Could not create database');
}

try {
    $result = $couch->dbDrop('wiki');
} catch (Zend_Couch_Exception $e) {
    die('Could not drop database');
}

$result = $couch->dbInfo('wiki');
echo<<<EOT
Document count: {$result->doc_count}
Database size: {$result->disk_size}
EOT;
```

UC-03

```
$couch = new Zend_Coucn('wiki');
$doc   = $couch->docOpen('WelcomePage');
foreach ($doc->toArray() as $field => $value) {
    echo "$field: " . var_export($value, 1) . "\n";
}
```

UC-04

```
$couch = new Zend_Coucn('wiki');
$docs  = $couch->allDocs();
foreach ($docs as $document) {
    echo $document->getId() . "\n";
}
```

UC-05

```
$couch = new Zend_Coucn('wiki');
$document = new Zend_Couch_Document();
$document->id = "NewPage";
$document->title = "New Page";
$document->content = "This is a new wiki page!"
$document->created = date('Y-m-d H:i:s');
$document->tags    = array('wiki', 'system');
$couch->docSave($document);
```

9. Class Skeletons

Please see my github repository for skeletons.

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