Zend_Service_Delicious - Goran Hacek

<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_Service_Delicious Component Proposal

Proposed Component Name	Zend_Service_Delicious
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Service_Delicious
Proposers	Goran Hacek
Revision	Jun 20, 2006 (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_Service_Delicious is a concrete implementation of the del.icio.us web service

2. References

- del.icio.us API documentation

3. Component Requirements, Constraints, and Acceptance Criteria

- Implement all API functions
- Handle authentication
- Prevent throttling

4. Dependencies on Other Framework Components

- Zend_Http_Client
- Zend_Service_Exception

5. Theory of Operation

Do a request, parse response and return data.

6. Milestones / Tasks

zone: Missing {zone-data:milestones}

7. Class Index

- Zend_Service_Delicious
- Zend_Service_Delicious_Post
- Zend_Service_Delicious_PostList
- Zend_Service_Delicious_Exception

8. Use Cases

```
$delicio = new Zend_Service_Delicious('uname', 'pass');

// download all posts
$allposts = $delicio->getAllPosts();

// change title of a post
$allposts[0]->setTitle('New title');
$allposts[0]->save();

// extract posts between 2 and 3 days of age with tag 'php'
// from downloaded post list
$olderPosts = $allposts->startDate(strtotime('-3 day'))
                       ->endDate(strtotime('-2 day'))
                       ->withTag('php');

// output posts
foreach ($olderPosts as $post) {
 output_post($post);
}
```

9. Class Skeletons

```
class Zend_Service_Delicious {
 const API_URI    = 'https://api.del.icio.us/';
 const PATH_UPDATE       = '/v1/posts/update';
 const PATH_TAGS         = '/v1/tags/get';
```

```
const PATH_TAG_RENAME   = '/v1/tags/rename';
const PATH_BUNDLES      = '/v1/tags/bundles/all';
const PATH_BUNDLE_DELETE= '/v1/tags/bundles/delete';
const PATH_BUNDLE_ADD   = '/v1/tags/bundles/set';
const PATH_DATES        = '/v1/posts/dates';
const PATH_POST_DELETE  = '/v1/posts/delete';
const PATH_POSTS_GET    = '/v1/posts/get';
const PATH_POSTS_ALL    = '/v1/posts/all';
const PATH_POSTS_ADD    = '/v1/posts/add';
const PATH_POSTS_RECENT = '/v1/posts/recent';

/**
 * Zend_Http_Client instance
 *
 * @var Zend_Http_Client
 */
protected $http;
/**
 * Microtime of last request
 *
 * @var float
 */
protected $lastRequestTime = 0;

/**
 * Constructs a new del.icio.us Web Services Client
 *
 * @param string $uname Clients username
 * @param string $pass  Clients password
 * @return Zend_Service_Delicious
 */
public function __construct($uname, $pass);
/**
 * Set clients username and password
 *
 * @param string $uname Clients user name
 * @param string $pass  Clients password
 * @return Zend_Service_Delicious
 */
public function setAuth ($uname, $pass);
/**
 * Get time of the last update
 *
 * @return int unix timestamp
 */
public function getLastUpdate ();
/**
 * Get all tags.
 * Returned array contains tags as keys and number of posts in tag as values
 *
 * @return array list of tags
 */
public function getTags ();
/**
 * Rename a tag
 *
 * @param string $old Old tag name
 * @param string $new New tag name
 */
public function renameTag ($old, $new);
/**
 * Get all bundles.
 * Returned array contains bundles as keys and list of tags as values.
 * Tags in lists are separated by spaces
 *
 * @return array list of tags
 */
public function getBundles ();
/**
 * Adds a new bundle
```

```
 *
 * @param string $bundle Name of new bundle
 * @param array $tags List of tags
 */
public function addBundle ($bundle, $tags);
/**
 * Delete a bundle
 *
 * @param string $bundle Name of bundle to be deleted
 */
public function deleteBundle ($bundle);
/**
 * Delete a post
 *
 * @param string $url URL of post to be deleted
 */
public function deletePost ($url);
/**
 * Get number of posts by date. Returns array where keys are dates
 * and values are numbers of posts.
 *
 * @param string $tag Optional filtering by tag
 * @return array list of dates
 */
public function getDates ($tag = null);
/**
 * Get posts matching the arguments. If no date or url is given, most recent date will be used.
 *
 * @param string $tag Optional filtering by tag
 * @param string $dt  Optional filtering by date
 * @param string $url Optional filtering by url
 * @return Zend_Service_Delicious_PostList
 */
public function getPosts ($tag = null, $dt = null, $url = null);
/**
 * Get all posts.
 *
 * @param string $tag Optional filtering by tag
 * @return Zend_Service_Delicious_PostList
 */
public function getAllPosts ($tag = null);
/**
 * Get recent posts.
 *
 * @param string $tag   Optional filtering by tag
 * @param string $count Maximal number of posts to be returned
 * @return Zend_Service_Delicious_PostList
 */
public function getRecentPosts ($tag = null, $count = 15);
/**
 * Handles all GET requests to a web service
 *
 * @param string $path Path
 * @param array $parms List of GET parameters
 * @return string response from web service
 */
protected function makeRequest ($path, $parms = array());
/**
 * Transform XML string to array
 *
 * @param string $response XML string to be parsed
 * @param string $root     Name of root tag
 * @param string $child    Name of children tags
 * @param string $attKey   Attribute of child tag to be used as a key
 * @param string $attValue Attribute of child tag to be used as a value
 * @return array
 */
protected static function responseToArray ($response, $root, $child, $attKey, $attValue);
/**
 * Builds array of posts from XML response
```

```
 *
 * @param string $response XML string to be parsed
 * @return array
 */
protected function parsePostList ($response);
/**
 * Evaluates response XML
 *
 * @param string $response XML string to be parsed
```

```
   */
  public static function evalResult ($response);
 }
```

```
class Zend_Service_Delicious_PostList implements Iterator, Countable, ArrayAccess {
 public function count();
 public function current();
 public function key();
 public function next();
 public function rewind();
 public function valid();

 public function offsetExists($offset);
 public function offsetGet($offset);
 public function offsetSet($offset, $value);
 public function offsetUnset($offset);

 /**
  * Constructs a new list of posts
  *
  * @param DOMNodelist $postsList
  * @return Zend_Service_Delicious_PostList
  */
 public function __construct(DOMNodelist $postsList);
 /**
  * Returns list of posts created after $date
  *
  * @parm int $date Unix timestamp
  * @return Zend_Service_Delicious_PostList
  */
 public function startDate($date);
 /**
  * Returns list of posts created before $date
  *
  * @parm int $date Unix timestamp
  * @return Zend_Service_Delicious_PostList
  */
 public function endDate($date);
 /**
  * Returns list of posts with tag $tag
  *
  * @parm string $tag Tag name
  * @return Zend_Service_Delicious_PostList
  */
 public function withTag($tag);
 /**
  * Returns list of posts with all tags from $tags
  *
  * @parm array $tags List of tags
  * @return Zend_Service_Delicious_PostList
  */
 public function withTags($tags);
 /**
  * Returns list of posts with URL that matches to the regular expression
  *
  * @parm string $regex
  * @return Zend_Service_Delicious_PostList
  */
 public function withURL($regex);
 /**
  * Returns list of posts with title that matches to the regular expression
  *
  * @parm string $regex
  * @return Zend_Service_Delicious_PostList
  */
```

```
public function withTitle($regex);
/**
 * Returns list of posts with title that matches to the regular expression
 *
 * @parm string $regex
 * @return Zend_Service_Delicious_PostList
```

```
   */
  public function withNote($regex);
 }
```

```
class Zend_Service_Delicious_Post {
 /**
  * Service that has downloaded the post
  *
  * @var ZServiceDelicio
  */
 protected $service;
 /**
  * Post content
  */
 protected $url;
 protected $title;
 protected $notes;
 protected $others;
 protected $tags;
 protected $time;
 protected $hash;

 /**
  * Constructs a new post
  *
  * @param Zend_Service_Delicious $service Service that has downloaded the post
  * @param array $content Post content
  * @return Zend_Service_Delicious_Post
  */
 public function __construct(Zend_Service_Delicious $service, $inArr);
 /**
  * Getter for title
  *
  * @return string
  */
 public function getTitle();
 /**
  * Setter for title
  *
  * @param string $newTitle
  */
 public function setTitle($newTitle);
 /**
  * Getter for notes
  *
  * @return string
  */
 public function getNotes();
 /**
  * Setter for notes
  *
  * @param string $newNotes
  */
 public function setNotes($newNotes);
 /**
  * Getter for tags
  *
  * @return array
  */
 public function getTags();
 /**
  * Setter for tags
  *
  * @param array $newTags
  */
```

```
public function setTags($newTags);
/**
 * Add tag to tag list
 *
 * @param string $newTag
 */
public function addTag($newTag);
/**
 *
 *
 * @param string $tag
 * @return bool
 */
public function hasTag ($tag);
/**
 * Removes a tag from tag list
 *
 * @param string $tag
 */
public function removeTag ($tag);
/**
 * Removes all tags
 *
 * @param string $tag
 */
public function removeTags ();
/**
 * Getter for time
 *
 * @return int Unix timestamp
 */
public function getTime ();
/**
 * Setter for time
 *
 * @param int $newTime Unix timestamp
 */
public function setTime ($newTime);
/**
 * Getter for others
 *
 * @return string
 */
public function getOthers ();
/**
 * Deletes post
 */
public function delete ();
/**
 * Saves post
 */
public function save ();
/**
 * Extracts content from the DOM element of a post
 * @return array
```

```
   */
  public static function parsePostNode (DOMElement $postNode);
 }
```

```
class Zend_Service_Delicious_Exception extends Zend_Service_Exception {
}
```

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