source: flynax-bridge / trunk / src / API.php

Last change on this file was 3000033, checked in by v1rustyle, 18 months ago

Bump to 2.2.0

Property svn:eol-style set to `native`

File size: 10.5 KB

```php
<?php

namespace Flynax\Plugins\FlynaxBridge;

use WP_Error;
use WP_REST_Response;
use WP_User;

/**
 * Class API
 *
 * @since 2.0.0
 *
 * @package Flynax\Plugins\FlynaxBridge
 */
class API
{
    /**
     * @var array $routes
     */
    public static $routes = [
        'handshake' => 'handShake',
        'fl-token' => 'saveFlToken',
        'status' => 'getConnectionStatus',
        'recent-posts' => 'getRecentPosts',
        'update-listings-cache' => 'updateListingsCache',
        'bridge-uninstalled' => 'afterBridgeUninstall',
        'register-user' => 'registerUser',
        'update-user' => 'updateUser',
        'validate-user' => 'validateUser',
        'update-password' => 'updatePassword',
        'delete-user' => 'deleteUser',
    ];

    /**
     * Load Route
     *
     * @param string $route
     *
     * @since 2.2.0 - Method changed to static
     * @since 2.1.0
     */
    public static function loadRoute($route = '')
    {
        if (!$route || !isset(self::$routes[$route])) {
            return;
        }
```

```
48
49          $method = self::$routes[$route];
50          self::$method();
51      }
52
53      /**
54       * Running after WordPress bridge plugin uninstalling process
55       *
56       * @since 2.2.0 - Method changed to static
57       */
58      public static function afterBridgeUninstall()
59      {
60          $self = new self();
61          $tokenFromRequest = sanitize_post($_REQUEST['wp_token']);
62
63          if (!$self->isValidToken($tokenFromRequest)) {
64              $response = new WP_Error('token-exchange-error', __('Invalid WP token', FlynaxBridge::PLUGIN_KEY));
65
66              print(json_encode($response));
67              return;
68          }
69
70          delete_option('flb_wp_token');
71          delete_option('flb_fl_token');
72          delete_option('flb_fl_url');
73          delete_option('flb_flynax_listings');
74
75          $response = new WP_REST_Response(array(
76              'message' => __('All tokens has been successfully removed', FlynaxBridge::PLUGIN_KEY),
77          ), 200);
78
79          print(json_encode($response));
80      }
81
82      /**
83       * Update cache off all widgets
84       *
85       * @since 2.2.0 - Method changed to static
86       */
87      public static function updateListingsCache()
88      {
89          Cache::updateFlListings();
90      }
91
92      /**
93       * Get connection status between WordPress bridge and FlynaxBridge plugins
94       *
95       * @since 2.2.0 - Method changed to static
96       */
97      public static function getConnectionStatus()
98      {
99          if (get_option('flb_fl_token') && get_option('flb_wp_token')) {
100             $response = new WP_REST_Response(array(
101                 'message' => __('Plugins are connected successfully', FlynaxBridge::PLUGIN_KEY),
102             ), 200);
103         } else {
104             $response = new WP_Error('status-message', __('Plugins are not
connected', FlynaxBridge::PLUGIN_KEY), 401);
105         }
106
107         print(json_encode($response));
108     }
109
```

```
    /**
     * Greetings method of the bridges
     *
     * @since 2.2.0 - Method changed to static
     */
    public static function handShake()
    {
        $self = new self();
        $token = $self->generateToken();
        $data = array(
            'token' => $token,
        );

        if ($token) {
            (new Events)->afterTokenGenerate($token);
            $response = new WP_REST_Response($data, 200);
            print(json_encode($response));
            return;
        }

        $response = new WP_Error('handshake-error', __('Handshake error', FlynaxBridge::PLUGIN_KEY));

        print(json_encode($response));
    }

    /**
     * Save WordPress bridge token, which is coming from Flynax
     *
     * @since 2.2.0 - Method changed to static
     */
    public static function saveFlToken()
    {
        $self = new self();

        $log = sprintf("\n%s:\n%s\n", date('Y.m.d H:i:s'), print_r($_REQUEST, true));
        file_put_contents('response.log', $log, FILE_APPEND);

        $tokenFromRequest = sanitize_post($_REQUEST['wp_token']);
        $flToken = sanitize_post($_REQUEST['fl_token']);
        $flUrl = sanitize_post($_REQUEST['fl_path']);

        if (!$self->isValidToken($tokenFromRequest)) {
            $response = new WP_Error('token-exchange-error', __('Invalid WP token', FlynaxBridge::PLUGIN_KEY));
            print(json_encode($response));
            return;
        }

        if (add_option('flb_fl_token', $flToken)) {
            add_option('flb_fl_url', $flUrl);

            $data = array(
                'message' => 'Token has been successfully exchanged',
            );

            $response = new WP_REST_Response($data, 200);
            print(json_encode($response));
            return;
        }

        $response = new WP_Error(
            'token-exchange-error',
            __(
                "I couldn't save your token. Maybe it is already exist?",
```

```
Line
173                 FlynaxBridge::PLUGIN_KEY
174             )
175         );
176         print(json_encode($response));
177         return;
178     }
179
180     /**
181      * Get recent posts from WordPress
182      *
183      * @since 2.2.0 - Method changed to static
184      */
185     public static function getRecentPosts()
186     {
187         $limit = sanitize_post($_REQUEST['limit']);
188         $args = array(
189             'numberposts' => $limit,
190             'orderby' => 'post_date',
191             'post_type' => 'post',
192             'post_status' => 'publish',
193         );
194
195         $posts = wp_get_recent_posts($args, ARRAY_A);
196         $resultPosts = array();
197         foreach ($posts as $post) {
198             $htmlStrippedContent = strip_tags($post['post_content'] ?: $post['post_excerpt']);
199             $sanitizedContent = preg_replace('/\[\/?et_pb.*?\]/', '', $htmlStrippedContent);
200             $sanitizedContent = trim(preg_replace('/\s+/', ' ', $htmlStrippedContent));
201             $sanitizedContent = wp_slash($sanitizedContent);
202
203             $post['post_title'] = wp_slash($post['post_title']);
204
205             $postInfo = array(
206                 'title' => $post['post_title'],
207                 'excerpt' => $sanitizedContent,
208                 'img' => get_the_post_thumbnail_url($post['ID'], 'thumbnail'),
209                 'post_date' => $post['post_date'],
210                 'url' => get_permalink($post['ID']),
211             );
212
213             if ($authorInfo = get_user_by('ID', $post['post_author'])) {
214                 $postInfo['author_username'] = $authorInfo->user_login;
215                 $postInfo['display_name'] = $authorInfo->display_name;
216             }
217
218             $resultPosts[] = $postInfo;
219         }
220
221         if (empty($resultPosts)) {
222             $response = new WP_Error(
223                 'posts-not-found',
224                 __(
225                     "There are no published posts",
226                     FlynaxBridge::PLUGIN_KEY
227                 ),
228                 404);
229         }
230
231         $response = new WP_REST_Response(array('data' => $resultPosts), 200);
232         print(json_encode($response));
233         return;
234     }
235
```

```
    /**
     * Does provided token from WordPress bridge plugin is valid
     *
     * @param string $token
     * @return bool
     */
    public function isValidToken($token)
    {
        return $token == $this->getOurToken();
    }

    /**
     * Get FlynaxBridge token
     *
     * @return string
     */
    public function getOurToken()
    {
        return get_option('flb_wp_token');
    }

    /**
     * Generate FlynaxBridge token
     *
     * @param int $length - Token length
     * @return string
     */
    public function generateToken($length = 32)
    {
        return bin2hex(random_bytes($length));
    }

    /**
     * Escape json string
     *
     * @since 2.1.0
     */
    public function escapeJsonString($value = '')
    {
        $escapers = array("\\");
        $replacements = array("\\\\");
        $result = str_replace($escapers, $replacements, $value);

        return $result;
    }

    /**
     * Register a new user
     *
     * @since 2.2.0 - Method changed to static
     * @since 2.1.0
     */
    public static function registerUser()
    {
        $username = $_REQUEST['username'];
        $password = $_REQUEST['password'];
        $email = $_REQUEST['email'];
        $type = 'author';
        $firstName = $_REQUEST['first_name'];
        $lastName = $_REQUEST['last_name'];

        if (username_exists($username) || email_exists($email)) {
            return;
```

```
        } else {
            $userdata = array(
                'user_pass' => $password,
                'user_login' => $username,
                'user_email' => $email,
            );

            $user_id = wp_insert_user($userdata);
            update_user_meta($user_id, "first_name", $firstName);
            update_user_meta($user_id, "last_name", $lastName);

            require_once '../../../wp-load.php';
            $user = new WP_User($user_id);
            $user->set_role($type);

            $out = array(
                'status' => 'OK',
                'wp_user_id' => $user_id,
            );

            print(json_encode($out));
        }
    }

    /**
     * Update user information
     *
     * @since 2.2.0 - Method changed to static
     * @since 2.1.0
     */
    public static function updateUser()
    {
        $userID = $_REQUEST['ID'];
        $userdata = array(
            'ID' => $userID,
            'user_email' => $_REQUEST['user_email'],
        );

        wp_update_user($userdata);

        $firstName = $_REQUEST['first_name'];
        $lastName = $_REQUEST['last_name'];

        if ($firstName) {
            update_user_meta($userID, "first_name", $firstName);
        }
        if ($lastName) {
            update_user_meta($userID, "last_name", $lastName);
        }
    }

    /**
     * Validate user information
     *
     * @since 2.2.0 - Method changed to static
     * @since 2.1.0
     */
    public static function validateUser()
    {
        $exists = false;
        if (email_exists($_REQUEST['user_email'])) {
            $exists = true;
        }
```

```
Line
362
363         print(json_encode(['exists' => $exists]));
364     }
365
366     /**
367      * Update user password
368      *
369      * @since 2.2.0 - Method changed to static
370      * @since 2.1.0
371      */
372     public static function updatePassword()
373     {
374         $password = $_REQUEST['password'];
375         $userID = $_REQUEST['wp_user_id'];
376
377         wp_set_password($password, $userID);
378     }
379
380     /**
381      * Delete user
382      *
383      * @since 2.2.0 - Method changed to static
384      * @since 2.1.0
385      */
386     public static function deleteUser()
387     {
388         require_once '../../../wp-load.php';
389         require_once ABSPATH . 'wp-admin/includes/admin.php';
390
391         $userID = $_REQUEST['wp_user_id'];
392
393         wp_delete_user($userID);
394     }
395
396     /**
397      * Register all WordPress REST API endpoints, which will be used by this plugin
398      *
399      * @deprecated 2.1.0
400      */
401     public function registerEndpoints()
402     {}
403 }
```

Developers

Get Involved

Learn

Showcase

Plugins

Themes

Patterns

WordCamp

WordPress.TV

BuddyPress

bbPress

WordPress.com

Matt

Privacy

Public Code