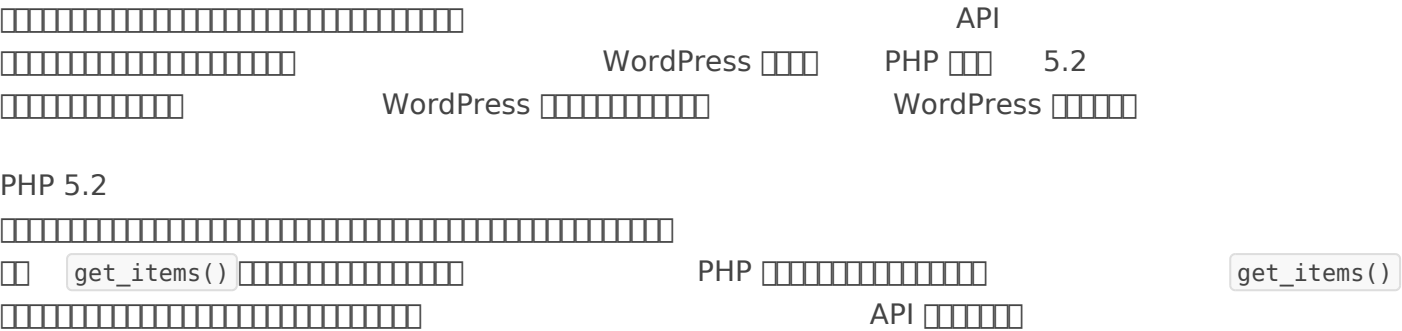




- [API](#)
- [Controllers](#)
- [WordPress REST API](#)



Controllers

Controllers typically do one thing; they receive input, and generate output. For the WordPress REST API our controllers will handle request input as `WP_REST_Request` objects and generate response output as `WP_REST_Response` objects. Let's look at an example controller class:

```
class My_REST_Posts_Controller {  
  
    // Here initialize our namespace and resource name.  
    public function __construct() {  
        $this->namespace = '/my-namespace/v1';  
        $this->resource_name = 'posts';  
    }  
}
```

```

    // Register our routes.
    public function register_routes() {
        register_rest_route( $this->namespace, '/' . $this->resource_name, array(
            // Here we register the readable endpoint for collections.
            array(
                'methods' => 'GET',
                'callback' => array( $this, 'get_items' ),
                'permission_callback' => array( $this, 'get_items_permissions_check' ),
            ),
            // Register our schema callback.
            array( $this, 'get_item_schema' ),
        ) );
        register_rest_route( $this->namespace, '/' . $this->resource_name . '/(?P<id>[\d]+)', array(
            // Notice how we are registering multiple endpoints the 'schema' equates to an OPTIONS
            request.
            array(
                'methods' => 'GET',
                'callback' => array( $this, 'get_item' ),
                'permission_callback' => array( $this, 'get_item_permissions_check' ),
            ),
            // Register our schema callback.
            array( $this, 'get_item_schema' ),
        ) );
    }

    /**
     * Check permissions for the posts.
     *
     * @param WP_REST_Request $request Current request.
     */
    public function get_items_permissions_check( $request ) {
        if ( ! current_user_can( 'read' ) ) {
            return new WP_Error( 'rest_forbidden', esc_html__( 'You cannot view the post resource.' ),
                array( 'status' => $this->authorization_status_code() ) );
        }
        return true;
    }

    /**

```

```

    * Grabs the five most recent posts and outputs them as a rest response.
    *
    * @param WP_REST_Request $request Current request.
    */
    public function get_items( $request ) {
        $args = array(
            'post_per_page' => 5,
        );
        $posts = get_posts( $args );

        $data = array();

        if ( empty( $posts ) ) {
            return rest_ensure_response( $data );
        }

        foreach ( $posts as $post ) {
            $response = $this->prepare_item_for_response( $post, $request );
            $data[] = $this->prepare_response_for_collection( $response );
        }

        // Return all of our comment response data.
        return rest_ensure_response( $data );
    }

    /**
     * Check permissions for the posts.
     *
     * @param WP_REST_Request $request Current request.
     */
    public function get_item_permissions_check( $request ) {
        if ( ! current_user_can( 'read' ) ) {
            return new WP_Error( 'rest_forbidden', esc_html__( 'You cannot view the post resource.' ),
                array( 'status' => $this->authorization_status_code() ) );
        }
        return true;
    }

    /**

```

```

    * Grabs the five most recent posts and outputs them as a rest response.
    *
    * @param WP_REST_Request $request Current request.
    */
    public function get_item( $request ) {
        $id = (int) $request['id'];
        $post = get_post( $id );

        if ( empty( $post ) ) {
            return rest_ensure_response( array() );
        }

        $response = prepare_item_for_response( $post );

        // Return all of our post response data.
        return $response;
    }

    /**
     * Matches the post data to the schema we want.
     *
     * @param WP_Post $post The comment object whose response is being prepared.
     */
    public function prepare_item_for_response( $post, $request ) {
        $post_data = array();

        $schema = $this->get_item_schema( $request );

        // We are also renaming the fields to more understandable names.
        if ( isset( $schema['properties']['id'] ) ) {
            $post_data['id'] = (int) $post->ID;
        }

        if ( isset( $schema['properties']['content'] ) ) {
            $post_data['content'] = apply_filters( 'the_content', $post->post_content, $post );
        }

        return rest_ensure_response( $post_data );
    }

```

```

/**
 * Prepare a response for inserting into a collection of responses.
 *
 * This is copied from WP_REST_Controller class in the WP REST API v2 plugin.
 *
 * @param WP_REST_Response $response Response object.
 * @return array Response data, ready for insertion into collection data.
 */
public function prepare_response_for_collection( $response ) {
    if ( ! ( $response instanceof WP_REST_Response ) ) {
        return $response;
    }

    $data = (array) $response->get_data();
    $server = rest_get_server();

    if ( method_exists( $server, 'get_compact_response_links' ) ) {
        $links = call_user_func( array( $server, 'get_compact_response_links' ), $response );
    } else {
        $links = call_user_func( array( $server, 'get_response_links' ), $response );
    }

    if ( ! empty( $links ) ) {
        $data['_links'] = $links;
    }

    return $data;
}

/**
 * Get our sample schema for a post.
 *
 * @param WP_REST_Request $request Current request.
 */
public function get_item_schema( $request ) {
    $schema = array(
        // This tells the spec of JSON Schema we are using which is draft 4.
        '$schema' => 'http://json-schema.org/draft-04/schema#',
    );

```

```

    // The title property marks the identity of the resource.
    'title'          => 'post',
    'type'           => 'object',
    // In JSON Schema you can specify object properties in the properties attribute.
    'properties'     => array(
        'id' => array(
            'description' => esc_html__( 'Unique identifier for the object.', 'my-textdomain' ),
            'type'        => 'integer',
            'context'     => array( 'view', 'edit', 'embed' ),
            'readonly'    => true,
        ),
        'content' => array(
            'description' => esc_html__( 'The content for the object.', 'my-textdomain' ),
            'type'        => 'string',
        ),
    ),
);

return $schema;
}

// Hook into the HTTP response
public function authorization_status_code() {

    $status = 401;

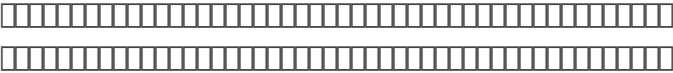
    if ( is_user_logged_in() ) {
        $status = 403;
    }

    return $status;
}

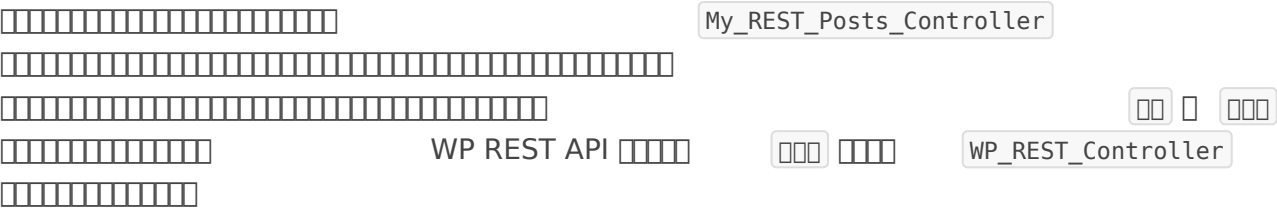
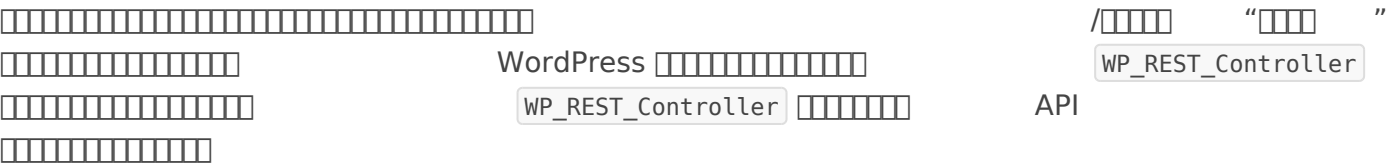
// Hook into the REST API
function prefix_register_my_rest_routes() {
    $controller = new My_REST_Posts_Controller();
    $controller->register_routes();
}

```

```
add_action( 'rest_api_init', 'prefix_register_my_rest_routes' );
```



```
class My_REST_Posts_Controller extends  
My_REST_Posts_Controller
```



#2	
2026-03-26 17:08:39 UTC	
2026-03-26 17:08:39 UTC	