Usage

Creating Routes

- Your plugin should hook into the **wp_router_generate_routes** action. The callback should take one argument, a **WP_Router** object.
- Register a route and its callback using **WP_Router::add_route($id, $args)**
    - **$id** is a unique string your plugin should use to identify the route
    - **$args** is an associative array, that sets the following properties for your route. Any omitted argument will use the default value.
        - **path** (required) - A regular expression to match against the request path. This corresponds to the array key you would use when creating rewrite rules for WordPress.
        - **query_vars** - An associative array, with the keys being query vars, and the values being explicit strings or integers corresponding to matches in the path regexp. Any query variables included here will be automatically registered.
        - **title** - The title of the page.
        - **title_callback** - A callback to use for dynamically generating the title. Defaults to __(). If NULL, the title argument will be used as-is. if page_callback or access_callback returns FALSE, title_callback will not be called.
        - **title_callback** can be either a single callback function or an array specifying callback functions for specific HTTP methods (e.g., GET, POST, PUT, DELETE, etc.). If the latter, the default key will be used if no other keys match the current request method.
        - **title_arguments** - An array of query variables whose values will be passed as arguments to title_callback. Defaults to the value of title. If an argument is not a registered query variable, it will be passed as-is.
        - **page_callback** (required) - A callback to use for dynamically generating the contents of the page. The callback should either echo or return the contents of the page (if both, the returned value will be appended to the echoed value). If FALSE is returned, nothing will be output, and control of the page contents will be handed back to WordPress. The callback will be called during the parse_request phase of WordPress's page load. If access_callback returns FALSE, page_callback will not be called.
        - **page_callback** can be either a single callback function or an array specifying callback functions for specific HTTP methods (e.g., GET, POST, PUT, DELETE, etc.). If the latter, the default key will be used if no other keys match the current request method.
        - **page_arguments** - An array of query variables whose values will be passed as arguments to page_callback. If an argument is not a registered query variable, it will be passed as-is.

- **access_callback** - A callback to determine if the user has permission to access this page. If access_arguments is provided, default is current_user_can, otherwise default is TRUE. If the callback returns FALSE, anonymous users are redirected to the login page, authenticated users get a 403 error.
- **access_callback** can be either a single callback function or an array specifying callback functions for specific HTTP methods (e.g., GET, POST, PUT, DELETE, etc.). If the latter, the default key will be used if no other keys match the current request method.
- **access_arguments** - An array of query variables whose values will be passed as arguments to access_callback. If an argument is not a registered query variable, it will be passed as-is.
- **template** - An array of templates that can be used to display the page. If a path is absolute, it will be used as-is; relative paths allow for overrides by the theme. The string $id will be replaced with the ID of the route. If no template is found, fallback templates are (in this order): route-$id.php, route.php, page-$id.php, page.php, index.php. If FALSE is given instead of an array, the page contents will be printed before calling exit() (you can also accomplish this by printing your output and exiting directly from your callback function).

Example:

```
$router->add_route('wp-router-sample', array(
    'path' => '^wp_router/(.*?)$',
    'query_vars' => array(
        'sample_argument' => 1,
    ),
    'page_callback' => array(get_class(), 'sample_callback'),
    'page_arguments' => array('sample_argument'),
    'access_callback' => TRUE,
    'title' => 'WP Router Sample Page',
    'template' => array('sample-page.php',
dirname(__FILE__).DIRECTORY_SEPARATOR.'sample-page.php')
));
```

In this example, the path http://example.com/wp_router/my_sample_path/ will call the function sample_callback in the calling class. The value of the sample_argument query variable, in this case "my_sample_path", will be provided as the first and only argument to the callback function. If the file sample-page.php is found in the theme, it will be used as the template, otherwise sample-page.php in your plugin directory will be used (if that's not found either, fall back to route-wp-router-sample.php, etc.).

Editing Routes

- You can hook into the wp_router_alter_routes action to modify routes created by other plugins. The callback should take one argument, a WP_Router object.

Public API Functions

Creating or changing routes should always occur in the context of the wp_router_generate_routes or wp_router_alter_routes actions, using the WP_Router object supplied to your callback function.

- WP_Router::edit_route(string $id, array $changes) - update each property given in $changes for the route with the given ID. Any properties not given in $changes will be left unaltered.
- WP_Router::remove_route(string $id) - delete the route with the given ID
- WP_Router::get_route(string $id) - get the WP_Route object for the given ID
- WP_Router::get_url(string $id, array $arguments) - get the URL to reach the route with the given ID, with the given query variables and their values
- WP_Route::get(string $property) - get the value of the specified property for the WP_Route instance.
