RESTful

Adding a generic REST adaptor to your module that uses the FreePBX API module for authentication and scopes. The generic rest adaptor follows and utilizes Slim v3 routes. To read more about slim v3 routes read this article https://www.slimframework.com/docs/v3/objects/router.html

In this example we are working with our Pony module that provides high service Ponies to your VoIP deployment.

The Ponies.php file should be placed under your module in the directory structure like so: <modulerawname>/Api/Rest/Ponies.php

Inside of the file we will include our generic base

```
<?php

namespace FreePBX\modules\Ponies\Api\Rest;
use FreePBX\modules\Api\Rest\Base;
class Ponies extends Base {
}
```

You've now created a generic Rest api class that does nothing! Congratulations!

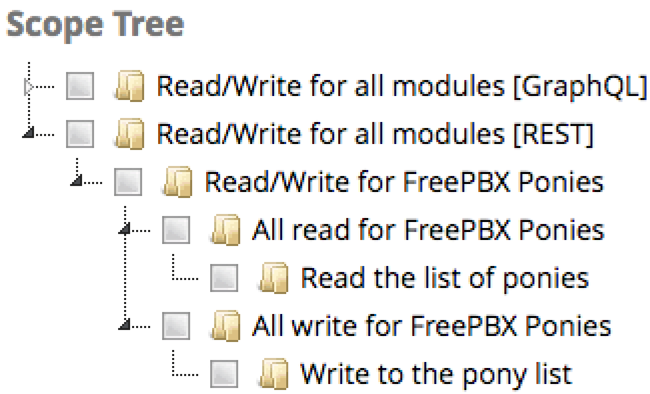
Next let's define some scopes for our module.

> You don't have to define any scopes and can instead use the catch all read and write for your module.

```
	public static function getScopes() {
		return [
			'read:ponylist' => [
				'description' => _('Read the list of ponies'),
			],
			'write:pnylist' => [
				'description' => _('Write to the pony list'),
			]
		];
	}
```

As you can see above we first define either read or write separated by a colon. You can ONLY define the words read or write in the first part of the scope. After that you can add anything you wish to help differentiate your scopes.

In the above example we've added a scope to read the pony list and write to the pony list. Save the file and refresh the scope visualizer in FreePBX



Now you see your two scopes listed in the scope tree for REST.

> When selecting scopes through the visualizer it will attempt to display the simplest scope. So if you click "Read the list of ponies" it will give you rest:read because there is only one item to read.

Now lets add the functional code

```
		public function setupRoutes($app) {
			$app->get('/list', function ($request, $response, $args) {
				$data = [];
				return $response->withJson($data);
			});
		}
```

The above example will create an API call like so: GET http://<hostname>/admin/api/api/rest/ponies/list

It will also return the response as JSON. However it is not checking scopes! To add scope checking you need to add middleware which is already provided by the underlying base class

This is a simple get request but you can use any route type provided by Slim (https://www.slimframework.com/docs/v3/objects/router.html)

```
		public function setupRoutes($app) {
			$app->get('/list', function ($request, $response, $args) {
				$data = [];
				return $response->withJson($data);
			});
		})->add($this->checkReadScopeMiddleware('ponylist'));
```

Now our middleware will check to make sure this call is listed in the valid scopes.

There are 5 different types of scope middleware

The first type will check to make sure you have a global read permissions such as rest:read

```
checkAllReadScopeMiddleware()
```

The second type will check to make sure you have a global write permissions such as rest:write

```
checkAllWriteScopeMiddleware()
```

The third type will check to make sure you have a read scope specific permissions such as rest:read:list

```
checkReadScopeMiddleware($scope)
```

The fourth type will check to make sure you have a write scope specific permissions such as rest:read:list

```
checkWriteScopeMiddleware($scope)
```

The fifth type will let you define the scope in full such as 'rest:read:list'

```
checkScopeMiddleware($scope)
```

Example

This is a working example in FreePBX currently under framework that provides system information

https://github.com/FreePBX/framework/blob/release/15.0/amp_conf/htdocs/admin/libraries/Api/Rest/System.php

http://<hostname>/admin/api/api/rest/framework/needreload

```php
<?php

namespace FreePBX\Api\Rest;
use FreePBX\modules\Api\Rest\Base;
class System extends Base {
	public static function getScopes() {
		return [
			'read:system' => [
				'description' => _('Read system
information'),
			]
		];
	}

	public function setupRoutes($app) {
		$app->get('/version', function ($request, $response,
$args) {
			$data = ['status' => true, 'version' =>
getVersion()];
			return $response->withJson($data);
		})->add($this->checkReadScopeMiddleware('system'));

		$app->get('/engine', function ($request, $response, $args)
{
			$data = ['status' => true, 'engine' =>
engine_getinfo()['version']];
			return $response->withJson($data);
		})->add($this->checkReadScopeMiddleware('system'));

		$app->get('/needreload', function ($request, $response,
$args) {
			$data = ['status' => true, 'needreload' =>
check_reload_needed()];
			return $response->withJson($data);
		})->add($this->checkReadScopeMiddleware('system'));
	}
}
```