**Toggle Navigation** 

## Setting up CORS

C Edit This Page (https://github.com/slimphp/Slim-Website/tree/gh-pages/docs/v4/cookbook/enable-cors.md)

CORS - Cross origin resource sharing

A good flowchart for implementing CORS support Reference:

CORS server flowchart

(http://www.html5rocks.com/static/images/cors server flowchart.png)

You can test your CORS Support here: http://www.test-cors.org/

You can read the specification here: https://www.w3.org/TR/cors/

## The simple solution

For simple CORS requests, the server only needs to add the following header to its response:

```
Access-Control-Allow-Origin: <domain>, ...
```

The following code should enable lazy CORS.

Add the following route as the last route:

```
<?php
use Slim\Exception\HttpNotFoundException;

/**

  * Catch-all route to serve a 404 Not Found page if none of the routes mat
  * NOTE: make sure this route is defined last
  */

$app->map(['GET', 'POST', 'PUT', 'DELETE', 'PATCH'], '/{routes:.+}', funct
    throw new HttpNotFoundException($request);
});
```

## Access-Control-Allow-Methods

The following middleware can be used to query Slim's router and get a list of methods a particular pattern implements.

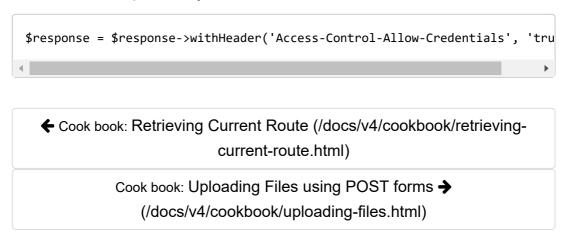
Here is a complete example application:

```
<?php
use Psr\Http\Message\ResponseInterface as Response;
use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Server\RequestHandlerInterface;
use Slim\Factory\AppFactory;
use Slim\Routing\RouteCollectorProxy;
use Slim\Routing\RouteContext;
require_once __DIR__ . '/../vendor/autoload.php';
$app = AppFactory::create();
$app->addBodyParsingMiddleware();
// This middleware will append the response header Access-Control-Allow-Me
$app->add(function (Request $request, RequestHandlerInterface $handler): R
    $routeContext = RouteContext::fromRequest($request);
    $routingResults = $routeContext->getRoutingResults();
    $methods = $routingResults->getAllowedMethods();
    $requestHeaders = $request->getHeaderLine('Access-Control-Request-Head
    $response = $handler->handle($request);
    $response = $response->withHeader('Access-Control-Allow-Origin', '*');
    $response = $response->withHeader('Access-Control-Allow-Methods', impl
    $response = $response->withHeader('Access-Control-Allow-Headers', $req
    // Optional: Allow Ajax CORS requests with Authorization header
    // $response = $response->withHeader('Access-Control-Allow-Credentials
    return $response;
});
// The RoutingMiddleware should be added after our CORS middleware so rout
$app->addRoutingMiddleware();
// The routes
$app->get('/api/v0/users', function (Request $request, Response $response)
    $response->getBody()->write('List all users');
    return $response;
});
$app->get('/api/v0/users/{id}', function (Request $request, Response $resp
    $userId = (int)$arguments['id'];
    $response->getBody()->write(sprintf('Get user: %s', $userId));
    return $response;
});
$app->post('/api/v0/users', function (Request $request, Response $response
    // Retrieve the JSON data
    $parameters = (array)$request->getParsedBody();
```

```
$response->getBody()->write('Create user');
    return $response;
});
$app->delete('/api/v0/users/{id}', function (Request $request, Response $r
    $userId = (int)$arguments['id'];
    $response->getBody()->write(sprintf('Delete user: %s', $userId));
    return $response;
});
// Allow preflight requests
// Due to the behaviour of browsers when sending a request,
// you must add the OPTIONS method. Read about preflight.
$app->options('/api/v0/users', function (Request $request, Response $respo
    // Do nothing here. Just return the response.
    return $response;
});
// Allow additional preflight requests
$app->options('/api/v0/users/{id}', function (Request $request, Response $
    return $response;
});
// Using groups
$app->group('/api/v0/users/{id:[0-9]+}', function (RouteCollectorProxy $gr
    $group->put('', function (Request $request, Response $response, array
        // Your code here...
        $userId = (int)$arguments['id'];
        $response->getBody()->write(sprintf('Put user: %s', $userId));
        return $response;
    });
    $group->patch('', function (Request $request, Response $response, arra
        $userId = (int)$arguments['id'];
        $response->getBody()->write(sprintf('Patch user: %s', $userId));
        return $response;
    });
    // Allow preflight requests
    $group->options('', function (Request $request, Response $response): R
        return $response;
    });
});
$app->run();
```

## Access-Control-Allow-Credentials

If the request contains credentials (cookies, authorization headers or TLS client certificates), you might need to add an Access-Control-Allow-Credentials header to the response object.



Created and maintained by

Josh Lockhart (http://joshlockhart.com), Andrew Smith (https://www.donielsmith.com), Rob Allen (http://akrabat.com/), Pierre Bérubé (http://www.lgse.com/), and the Slim Framework Team (https://github.com/orgs/slimphp/people)

