TABLE OF CONTENT

|  |  |  |
| --- | --- | --- |
| I | CODE OF RELATED FILES |  |
| II | SCREEN SHORT OF THE RESPONSES |  |
| III | YOUR GIT PROJECT REPOSITORY ADDRESS |  |

1. CODE OF RELATED FILES

**product.php**

<?php

use Slim\Http\Request; //namespace

use Slim\Http\Response; //namespace

//include productsProc.php file

include \_\_DIR\_\_ . '/../function/productsProc.php';

//read table products

$app->get('/products',

function (Request $request, Response $response, array $arg)

{

 $data = getAllProducts($this->db);

if(is\_null($data))

{

    return $this->response->withJson(array('error' => 'no data'), 404);

}

return $this->response->withJson(array('data' => $data), 200);

});

//request table products by condition

$app->get('/products/[{id}]', function ($request, $response, $args){

    $productId = $args['id'];

   if (!is\_numeric($productId)) {

      return $this->response->withJson(array('error' => 'numeric paremeter required'), 422);

   }

  $data = getProduct($this->db,$productId);

  if (empty($data)) {

    return $this->response->withJson(array('error' => 'no data'), 404);

 }

   return $this->response->withJson(array('data' => $data), 200);

});

//create new products

$app->post ('/products/new', function ($request, $response, $args)

{

   $form\_data = $request->getParsedBody();

   //return $this->response->withJson($form\_data, 200);

   $data = createProduct($this->db, $form\_data);

      if ($data <= 0)

      {

         return $this->response->withJson(array('error' => 'fail'), 500);

      }

   return $this->response->withJson(array('data' => 'success'), 200);

});

//edit products

$app->put('/products/edit/[{id}]', function ($request,  $response,  $args)

{

   $productId = $args['id'];

   $date = date("Y-m-j h:i:s");

  if (!is\_numeric($productId))

{

     return $this->response->withJson(array('error' => 'numeric paremeter required'), 422);

}

   $form\_data=$request->getParsedBody();

 $data=editProduct($this->db,$form\_data,$productId,$date);

 if ($data <= 0)

return $this->response->withJson(array('data' => 'success'), 200);

});

 //delete products

 $app->delete('/products/delete/[{id}]', function ($request, $response, $args)

 {

   $productId = $args['id'];

  if (!is\_numeric($productId))

  {

     return $this->response->withJson(array('error' => 'numeric paremeter required'), 422);

  }

 $data = deleteProduct($this->db,$productId);

 if (empty($data)) {

  return $this->response->withJson(array('data' => 'success'), 202);

 };

 });

**productsProc.php**

<?php

//get all products

function getAllProducts($db)

{

$sql = 'Select p.name, p.description, p.price, c.name as category from products p ';

$sql .='Inner Join categories c on p.category\_id = c.id';

$stmt = $db->prepare ($sql);

$stmt ->execute();

return $stmt->fetchAll(PDO::FETCH\_ASSOC);

}

//get product by id

function getProduct($db, $productId)

{

$sql = 'Select p.name, p.description, p.price, c.name as category from products p ';

$sql .= 'Inner Join categories c on p.category\_id = c.id ';

$sql .= 'Where p.id = :id';

$stmt = $db->prepare ($sql);

$id = (int) $productId;

$stmt->bindParam(':id', $id, PDO::PARAM\_INT);

$stmt->execute();

return $stmt->fetchAll(PDO::FETCH\_ASSOC);

}

//insert new product

function createProduct($db, $form\_data)

{

$sql = 'insert into products (name, description, price, category\_id, created)';

$sql .= 'values (:name, :description, :price, :category\_id, :created)';

$stmt = $db->prepare ($sql);

$stmt->bindParam(':name', $form\_data['name']);

$stmt->bindParam(':description', $form\_data['description']);

$stmt->bindParam(':price', floatval($form\_data['price']));

$stmt->bindParam(':category\_id', intval($form\_data['category\_id']));

$stmt->bindParam(':created', $form\_data['created']);

$stmt->execute();

return $db->lastInsertID();//insert last number.. continue

}

//edit product

function editProduct($db,$form\_data,$productId,$date)

{

$sql = 'UPDATE products SET name = :name , description = :description , price = :price , category\_id = :category\_id , modified = :modified ';

$sql .=' WHERE id = :id';

$stmt = $db->prepare ($sql);

$id = (int)$productId;

$mod = $date;

$stmt->bindParam(':id', $id, PDO::PARAM\_INT);

$stmt->bindParam(':name', $form\_data['name']);

$stmt->bindParam(':description', $form\_data['description']);

$stmt->bindParam(':price', floatval($form\_data['price']));

$stmt->bindParam(':category\_id', intval($form\_data['category\_id']));

$stmt->bindParam(':modified', $mod , PDO::PARAM\_STR);

$stmt->execute();

}

//delete products

function deleteProduct($db,$productId)

{

$sql = 'Delete from products where id = :id';

$stmt = $db->prepare($sql);

$id = (int)$productId;

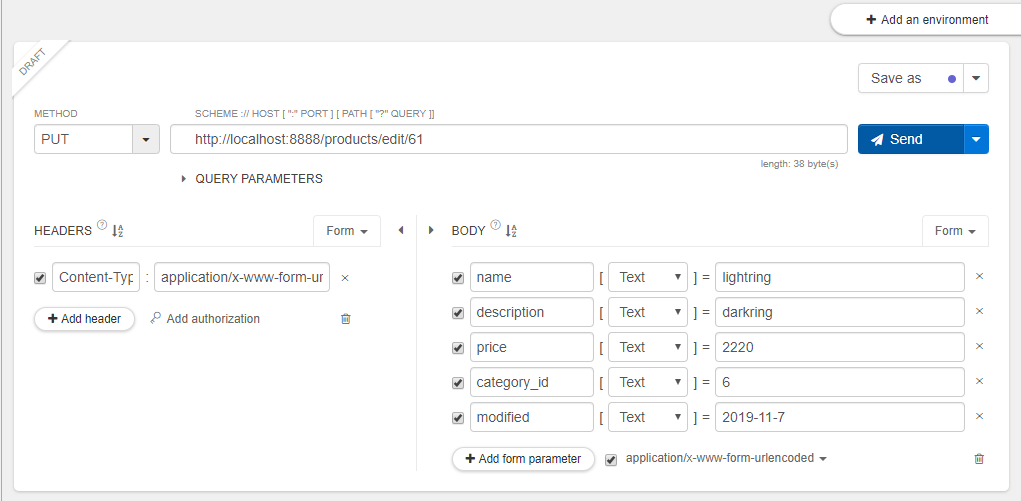
$stmt->bindParam(':id', $id, PDO::PARAM\_INT);

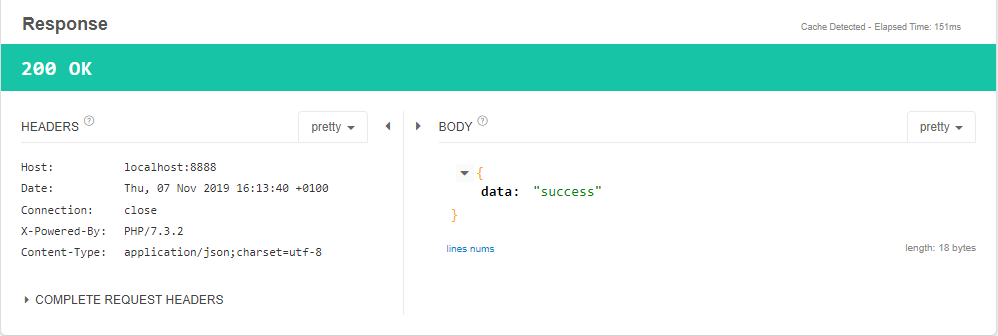
$stmt->execute();

}

1. SCREEN SHORT OF THE RESPONSES

PUT METHOD





DELETE METHOD

