Web Services Programming

DWES UD7.2

5.- XML file generation

An XML file can be generated in several ways:

- Simple XML
- Simple XML + Fileputcontents
- DOM

We are going to see SimpleXML

5.- XML file generation

```
<?php
    $xml = new SimpleXMLElement('<canciones/>');

    for ($i = 1; $i <= 8; ++$i) {
        $cancion = $xml->addChild('cancion');
        $cancion->addChild('ruta', "cancion$i.mp3");
        $cancion->addChild('titulo', "Cancion $i - Titulo");
    }

    $xml->asXML('canciones.xml’);
?>
```

6.- JSON file generation

There are mainly two ways to generate json:

- Json_encode
- Serialize

The recommended method that we are going to see is json_encode

6.- JSON file generation

- This code:

```
<?php
    $arr = array('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5);
    echo json_encode($arr);
?>
```

- Will have as result:

```
{"a":1,"b":2,"c":3,"d":4,"e":5}
```

6.- JSON file generation

An example with encoding options:

```
<?php
$a = array('<foo>','"bar"','"baz"','&blong&', "\xc3\xa9");
echo "Normal: ", json_encode($a), "\n";
echo "Tags: ", json_encode($a, JSON_HEX_TAG), "\n";
echo "Apos: ", json_encode($a, JSON_HEX_APOS), "\n";
echo "Quot: ", json_encode($a, JSON_HEX_QUOT), "\n";
echo "Amp: ", json_encode($a, JSON_HEX_AMP), "\n";
echo "Unicode: ", json_encode($a, JSON_UNESCAPED_UNICODE), "\n";
echo "All: ", json_encode($a, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_QUOT |
JSON_HEX_AMP | JSON_UNESCAPED_UNICODE), "\n\n";
```

6.- JSON file generation

```
$b = array();
echo "Array vacío retornado como array: ", json_encode($b), "\n";
echo "Array vacío retornado como object: ", json_encode($b, JSON_FORCE_OBJECT),
"\n\n";

$c = array(array(1,2,3));
echo "Array no asociativo retornado como array: ", json_encode($c), "\n";
echo "Array no asociativo retornado como objeto: ", json_encode($c,
JSON_FORCE_OBJECT), "\n\n";

$d = array('foo' => 'bar', 'baz' => 'long');
echo "Array asociativo siempre es retornado como objeto: ", json_encode($d), "\n";
echo "Array asociativo siempre es retornado como objeto: ", json_encode($d,
JSON_FORCE_OBJECT), "\n\n";
?>
```

6.- JSON file generation

The result of the previous code would be:

Normal: ["<foo>","'bar'","\"baz\"","&blong&","\u00e9"]
Tags: ["\u003Cfoo\u003E","'bar'","\"baz\"","&blong&","\u00e9"]
Apos: ["<foo>","\u0027bar\u0027","\"baz\"","&blong&","\u00e9"]
Quot: ["<foo>","'bar'","\u0022baz\u0022","&blong&","\u00e9"]
Amp: ["<foo>","'bar'","\"baz\"","\u0026blong\u0026","\u00e9"]
Unicode: ["<foo>","'bar'","\"baz\"","&blong&","é"]
All: ["\u003Cfoo\u003E","\u0027bar\u0027","\u0022baz\u0022","\u0026blong\u0026","é"]

Array vacío retornado como array: []
Array vacío retornado como object: {}

Array no asociativo retornado como array: [[1,2,3]]
Array no asociativo retornado como objeto: {"0":{"0":1,"1":2,"2":3}}

Array asociativo siempre es retornado como objeto: {"foo":"bar","baz":"long"}
Array asociativo siempre es retornado como objeto: {"foo":"bar","baz":"long"}

Exercises

- Check all previous examples
- Read and understand all decode options in the documentation

7 - RESTful services creation

We will only see how to create a restful service with json. We will start with this script:

```
<?php
  $arr = array('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5);
  header("Content-Type:application/json");
  echo json_encode($arr);
?>
```

7 - RESTful services creation

We can add operations to our service using:

```
$_SERVER['REQUEST_METHOD'] == 'GET’   //(SELECT)
$_SERVER['REQUEST_METHOD'] == 'POST’ //(INSERT)
$_SERVER['REQUEST_METHOD'] == 'DELETE’
$_SERVER['REQUEST_METHOD'] == 'PUT’   //(UPDATE)
```

Do we want everyone to be able to modify our data?

7 - RESTful services creation

We modify our service leaving the following:

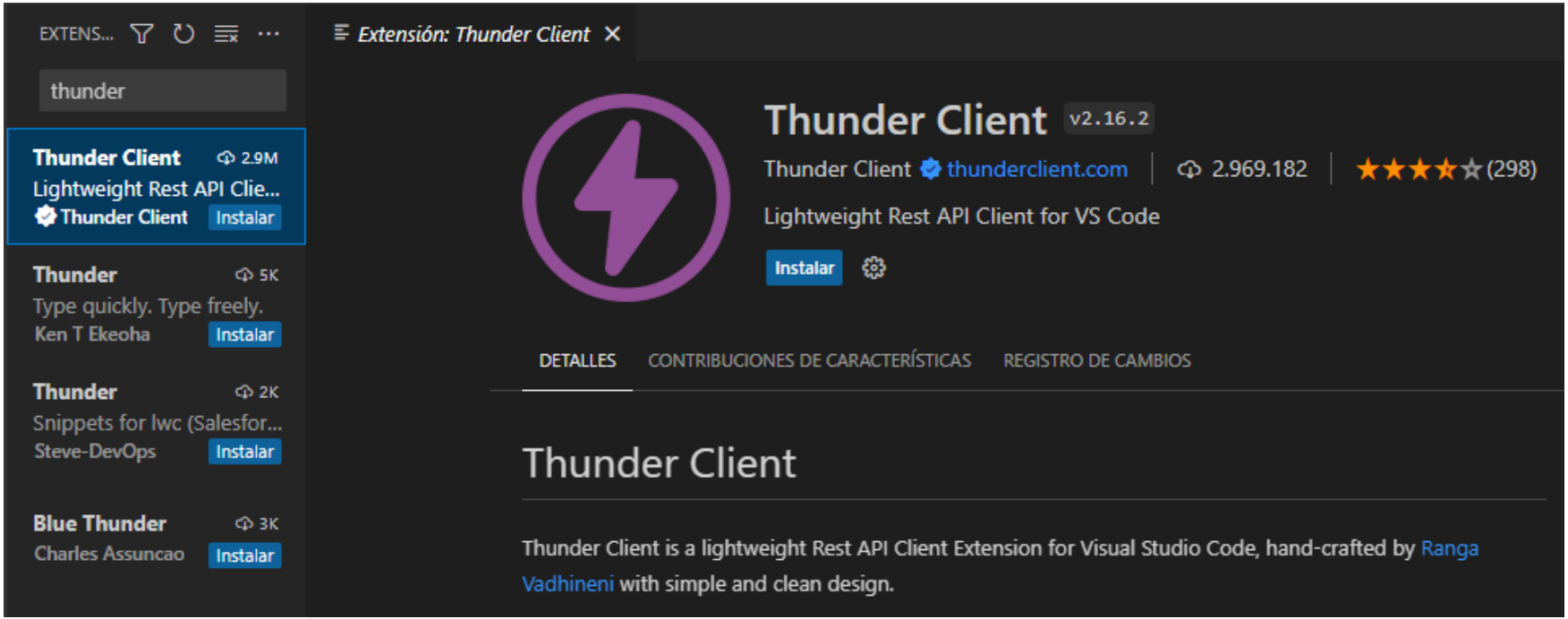
```
$arr = array('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5);
switch ($_SERVER["REQUEST_METHOD"]){
  case "PUT":
     echo "Estoy haciendo UPDATE";
     break;
  case "POST":
     echo "Estoy haciendo INSERT";
     break;
```

7 - RESTful services creation

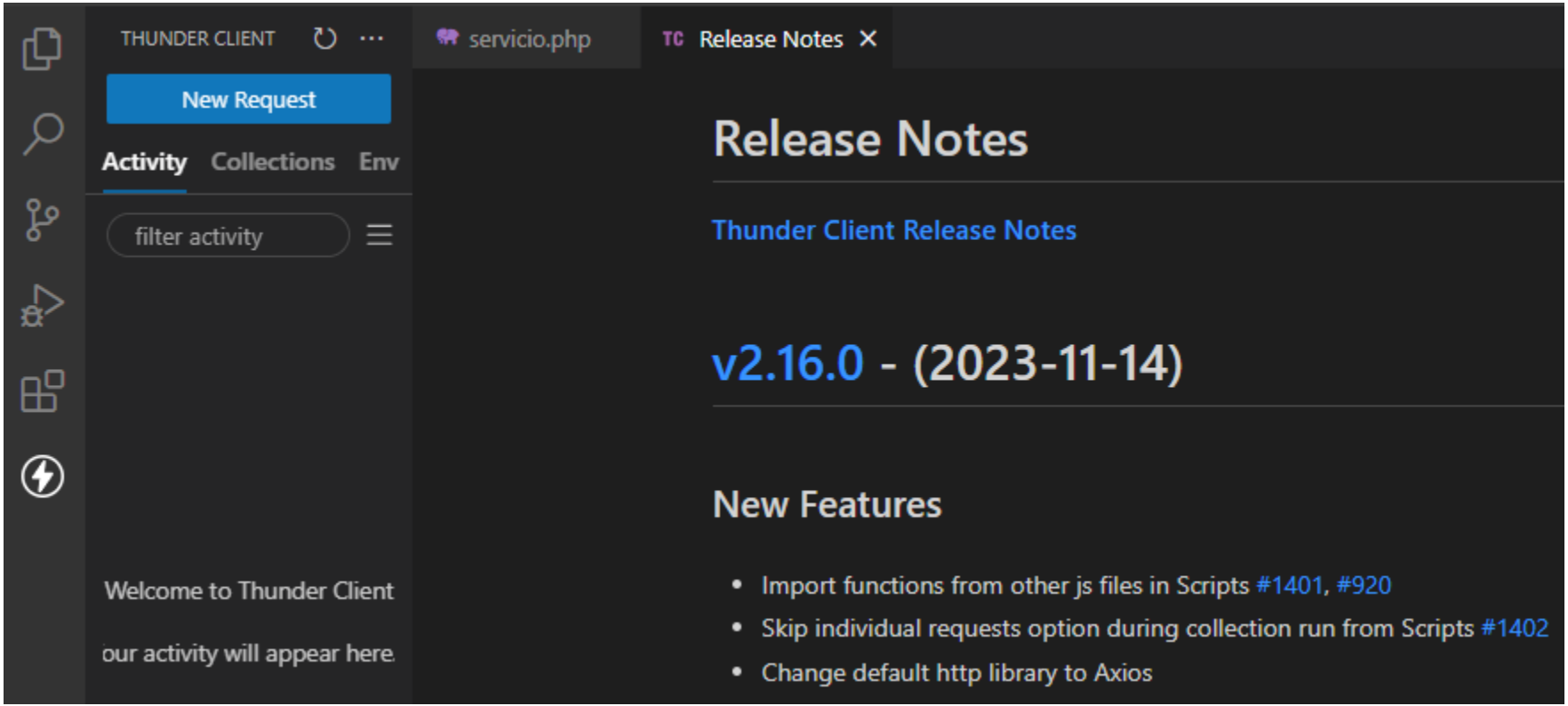
```
case "DELETE";
      echo "Estoy haciendo DELETE";
      break;
  case "GET":
  default:
     header("Content-Type:application/json");
     echo json_encode($arr);
     break;
 }
```

7 - RESTful services creation

To test our service we can use Thunder Client:

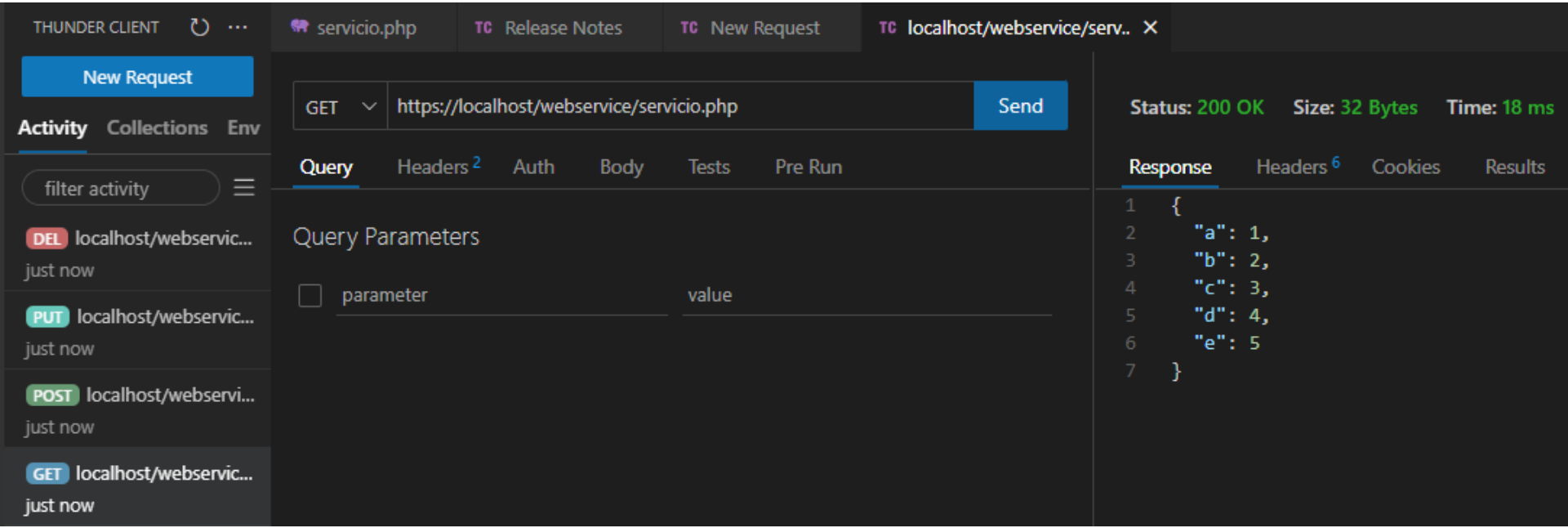


7 - RESTful services creation



7 - RESTful services creation

We can test all the operations easily:



7 - RESTful services creation

We already know how to create a web service. Now we are going to create a service to manage students. First, we create a ‘webservice’ database, then a table:

```
CREATE TABLE IF NOT EXISTS `alumnos` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `nombre` varchar(30) COLLATE utf8_spanish_ci NOT NULL,
  `apellido1` varchar(30) COLLATE utf8_spanish_ci NOT NULL,
  `apellido2` varchar(30) COLLATE utf8_spanish_ci,
  `telefono` varchar(30) COLLATE utf8_spanish_ci NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;
```

7 - RESTful services creation

We create the database user that will use our service, granting only select, insert, update and delete privileges:

```
CREATE USER 'webservice'@'%' IDENTIFIED BY 'webservice';
GRANT SELECT, INSERT, UPDATE, DELETE ON *.* TO 'webservice'@'%' REQUIRE NONE
WITH MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0
MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0;
```

7 - RESTful services creation

Now, we store database credentials in a file:

```
{"host":"localhost","username":"webservice","password":"webservice",“db":"webservice"}
```

In our webservice we read the credentials:

```
$db = json_decode(file_get_contents ('credenciales.txt’),true);
```

What type of variable is $db?

7 - RESTful services creation

The function to connect to the DB will be:

```
function conectarBd($db)
{
    try {
        $conn = new PDO("mysql:host={$db['host']};dbname={$db['db']};charset=utf8",
$db['username'], $db['password']);
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        return $conn;
    } catch (PDOException $exception) {
        exit($exception->getMessage());
    }
}
$dbCon =  conectarBd($db);
```

7 - RESTful services creation

```
if ($_SERVER['REQUEST_METHOD'] == 'GET') {
        $sql = "SELECT * from alumnos";
        $resultado = $dbCon->query($sql);
        $miArrayPrin = array();
        $miArray = array();
        while ($registro = $resultado->fetch()) {
                $miArray["nombre"]=$registro["nombre"];
                $miArray["apellido1"]=$registro["apellido1"];
                $miArray["apellido2"]=$registro["apellido2"];
                $miArray["telefono"]=$registro["telefono"];

                $miArrayPrin[$registro["id"]] = $miArray;
        }
        echo json_encode($miArrayPrin);
}
```

7 - RESTful services creation

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $input =json_decode(file_get_contents('php://input'), true);
    $sql = "INSERT INTO alumnos (nombre, apellido1, apellido2, telefono)
          VALUES
          ('{$input['nombre']}', '{$input['apellido1']}', '{$input['apellido2']}', '{$input['telefono']}')";
    echo $sql;
    $dbCon->exec($sql);
    $aluId = $dbCon->lastInsertId();
    if($aluId) {
      $input['id'] = $aluId;
      header("HTTP/1.1 200 OK");
      echo json_encode($input);
      exit();
                }
}
```

7 - RESTful services creation

```
if ($_SERVER['REQUEST_METHOD'] == 'DELETE')
{
  $input =json_decode(file_get_contents('php://input'), true);
  $sql = $dbCon->prepare("DELETE FROM alumnos where id=:id");
  $sql->bindValue(':id', $input['id']);
  $sql->execute();
  header("HTTP/1.1 200 OK");
  exit();
}
```

7 - RESTful services creation

```
if ($_SERVER['REQUEST_METHOD'] == 'PUT') {
    $input = json_decode(file_get_contents('php://input'), true);
    $postId = $input['id'];
    $telefono = $input['telefono'];
    $sql = "
        UPDATE alumnos
        SET telefono='{$telefono}'
        WHERE id={$postId}
         ";
    echo $sql;
    $dbCon->exec($sql);
    header("HTTP/1.1 200 OK");
    exit();
}
```

7 - Exercise

- In groups, create a RESTful API to maintain hotel reservations.
- Design and create the database with the fields considered necessary.
- The API must allow querying, adding, deleting and modifying reservations through web services, limiting access, so that only authorized users can modify data.
- The delivery will consist of uploading to Aules the link to the GitHub repository that contains everything necessary to deploy the application in a new environment and for it to work correctly, that is at least, the application source code and the SQL files necessary to create and populate the database.

7 – Exercise

Groups:

- Jose Miguel ,Pablo y Belén
- Nicolás y Pedro
- Elizabeth y Alberto
- Minda y Alí
- Alex y Gabi
- Carlos e Ismael
- Adrián y Lucía
- Gian Carlos y Jordi
- Joel y Juan

Tip: plan your work, coordinate, distribute tasks and share the Github repository