APLICACIONES

REST

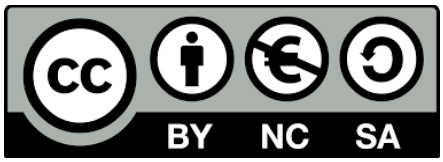
con validación por

TOKEN JWT

Antonio Boronat Pérez

Desarrollo de Aplic. Web Servidor

CFGS DAW



Índice

Introducción

Las aplicaciones REST se consideran sin estado, cada petición es independiente de las anteriores y no se guarda ninguna información en el servidor sobre ellas, a diferencia del caso en que se usan sesiones en las que el servidor conserva datos sobre la evolución del intercambio de información con el cliente.

Visto el escenario de funcionamiento de las aplicaciones REST, podemos necesitar en algún momento de un mecanismo que pueda conservar información entre llamadas al servidor, por ejemplo cuando nos debemos validar como usuarios y no vamos a realizar esta operación en cada petición al servidor. Para esto usaremos la librería de token JWT (JSON WEB TOKEN). Este elemento va cifrado para proteger su información y se envía al servidor en las cabeceras de la petición Web.

JWT es un estándar para realizar peticiones de forma segura entre dos elementos siguiendo el RFC7519, intercambiando información en formato JSON. Esta información va firmada , bien con una contraseña HMAC o con certificados ECDSA o RSA. JWT puede usarse para dos cometidos: autenticación e intercambio de información.

En este documento, nos centraremos en el uso de autenticación, de forma que una vez un usuario se ha validado, se genera un token que se enviará en las siguientes peticiones y servirá para permitir el uso de la aplicación.

Instalación JWT - FIREBASE

Usaremos las librerías ofrecidas por FIREBASE para PHP. Las instalación la realizamos usando la utilidad *composer* en el directorio raíz del proyecto:

```
composer require firebase/php-jwt
```

El siguiente paso consistirá en configurar las clases que manejan los token JWT en nuestro proyecto. Debemos añadir la trayectoria al directorio de *Firebase,* dentro del directorio del proyecto: `vendor/firebase/php-jwt/src` y pondremos con `include` los ficheros que se muestran en el código de ejemplo.

Además, el fichero JWT.php tiene la clase JWT y define el namespace `Firebase\JWT` que tendremos que usar en la llamada a los métodos que gestionan los token.

Uso de JWT

Básicamente, necesitamos las operaciones para crear un token y para comprobar su contenido. Facilita estas operaciones usar una clase que incorpore los métodos necesarios, en nuestro caso, la clase *Authentication*[1]. Esta clase está en el fichero `Authentication.php` en el directorio de nuestro proyecto de ejemplo.

Authentication.php:

```
<?php

include "vendor/firebase/php-jwt/src/JWTExceptionWithPayloadInterface.php";
include "vendor/firebase/php-jwt/src/SignatureInvalidException.php";
//include "vendor/firebase/php-jwt/src/ExpiredException.php";
include "vendor/firebase/php-jwt/src/JWT.php";
include "vendor/firebase/php-jwt/src/Key.php";
class Authentication {
private $decodedToken = null;

private $jwtKey = "123.IESjc.zzz";

public function getDecodedToken() {
  return $this->decodedToken;
}
/**
* Valida que en la cabecera HTTP "Authorization" haya un token válido
* @return Mensaje de error (vacío si todo ha ido bien)
*/
public function validaToken() {
  $error = '';
  if(!array_key_exists('Authorization',getallheaders())) { // Comprueba la cabecera
     $error = 'No se ha iniciado sesión en la aplicación';
  }
  else {
    $authorization = getallheaders()['Authorization']; // Extrae le token de la cabecera
    $trozos = explode(' ', $authorization);
    $auth = $trozos[1]; // Normalmente recibimos 'Bearer token'
```

1 Obtenida del curso del CEFIRE Desarrollo de aplicaciones web MVC en el servidor con elframework PHP: Phalcon del profesor Arturo Bernal Mayordomo

```
   try {
      $decoded = \Firebase\JWT\JWT::decode($auth, new \Firebase\JWT\Key($this->jwtKey, 'HS256'));
   } catch(\Firebase\JWT\SignatureInvalidException $e) {
      $error = 'No se ha iniciado sesión en la aplicación';
   } //catch(\Firebase\JWT\ExpiredException $e) {
   catch(Exception $e) {
      $error = 'Excepcion: sesión caducada';
   }
  }
  if($error === '') {
    $this->decodedToken = $decoded;
  }
  return $error;
} // validaToken()
/**
* Genera un token para la autenticación del usuario en la aplicación
* @param $usuario Objeto del modelo con los datos del usuario autenticado
* @return Token generado con JWT. Caduca al mes.
*/
public function generaToken($usuario, $rol='profe') {
  $tiempo = time();
  $token = \Firebase\JWT\JWT::encode(
    [ // Contenido del token
      'exp' => $tiempo + 60, // 1 minuto y 60*60 --> Caduca en una hora
      'id' => $usuario,
      'rol' => $rol,
    ],
   $this->jwtKey, // Clave JWT
   'HS256' // Algoritmo de codificación del token
  );
  return $token;
}
} class Authentication
?>
```

Ahora veamos cómo usar esta clase en nuestra aplicación:

Empezamos por el primer paso que será que la aplicación cliente necesite validar un usuario y crear un token para él (`cliente.php`) y llamará al servidor de la aplicación REST (`index.php`):

```
<?php
/* Cliente REST. Como ejemplo, pide la creación del token, si va bien pasa a bienvenida.php y también puede comprobar si es válido, si no ha caducado. En este ejemplo se crea con un minuto de duración.
*/

// Crea una sesión
session_start();
// Si se valida correctamente, contendrá el valor introducido en el form.
if(!isset( $_SESSION["token"])){
   $_SESSION["token"] = "";
}
?>
<!DOCTYPE html>
<!-- Ejemplo para validar usurios aplicacion REST y el creat TOKEN JWT -->
<html>
<head><meta charset="UTF-8"></head>
<body>
<form name="autenticar" method="post" action="<?php echo
htmlspecialchars($_SERVER['PHP_SELF']); ?>" >
<label>Usuario: </label> <input type="text" name="usuario"><br>
<label>Contraseña: </label> <input type="password" name="passwd"><br>
<input type="submit" name="validar" value="Login">
<input type="submit" name="validartoken" value="Validar Token">
<input type="reset" name="cancelar" value="Cancelar">
</form>

<?php
// Formulario para enviar al servidor REST
include("curl_conexion.php");
include("config.php");
$error_validacion= NULL;
if(isset($_REQUEST["validar"])){
   if(isset($_REQUEST["usuario"]) && isset($_REQUEST["passwd"])){
      $usuario = filter_input(INPUT_POST, "usuario", FILTER_SANITIZE_STRING);
      $passwd = filter_input(INPUT_POST, "passwd", FILTER_SANITIZE_STRING);
      $passwd = password_hash($passwd, PASSWORD_BCRYPT);//Encripta el password
      $url = _URL_SERVIDOR_ . "index.php?usuario=" . $usuario . "&passwd=" . $passwd;
      $response = curl_conexion($url, "GET");
      $resp = json_decode($response);
      if($resp->status === "200"){
         $_SESSION["token"] = $resp->valor;
         header("Location: bienvenida.php");
```

```
        exit;
    }
    else{
      $error_validacion= $resp->valor . "<br>";
      echo $error_validacion;
    }
    } // usuario y passwd
} // validar
// Comprueba si el token es válido y puede recuperar la información contenida
if(isset($_REQUEST["validartoken"])){
    $url = _URL_SERVIDOR_ . "index.php?validarToken=" . "valida";
    $response = curl_conexion($url, "GET", $_SESSION["token"]);
    $resp = json_decode($response);
    if($resp->status === "200"){
       header("Location: bienvenida.php");
       exit;
    }
    else{
      $error_validacion= $resp->status . " Error: " . $resp->valor . "<br>";
      echo $error_validacion;
    }
} // validarToken()
?>
</body>
</html>
```

El código del servidor (`index.php`) además de las funciones propias de la aplicación, crea o comprueba el token usando la clase *Authentication* :

```
<?php

/*
* Aplicación de validación y crea token JWT servicios Web REST
* Servidor REST
* Los usuarios y contraseñas están en un array en config.php
*/
include 'config.php'; // Tiene un array con usuarios $credenciales
include 'Authentication.php'; // Clase para gestionar los token JWT

$metodo = $_SERVER['REQUEST_METHOD'];

$auth = new Authentication();
```

```
switch($metodo){
case 'GET':
  if(isset($_REQUEST['usuario']) && isset($_REQUEST['passwd'])){
    $usuario = filter_input(INPUT_GET, 'usuario',FILTER_SANITIZE_STRING);
    $passwd = filter_input(INPUT_GET, 'passwd',FILTER_SANITIZE_STRING);
    if(array_key_exists($usuario, $credenciales)) {
       // Comprueba el password encriptado
       if(password_verify($credenciales[$usuario], $passwd)){
         $token = $auth->generaToken($usuario);
         $con_bd = ["status" => "200", "valor" => $token];
       }
       else {
        $con_bd = ["status" => "401", "valor" => "Password incorrecto"];
       }
    }
    else {
        $con_bd = ["status" => "401", "valor" => "Usuario incorrecto"];
    }
    echo json_encode($con_bd, TRUE);
  } //if(isset

  if(isset($_REQUEST['validarToken'])){ // Obtiene el dato del token
    $error = $auth->validaToken();
    if(strlen($error) > 0) { // Token error
       $res = ["status" => "401", "valor" => "No está autenticado - " . $error];
    }
    else{
      $usu = $auth->getDecodedToken(); // Toma los datos del token
      $res = ["status" => "200", "valor" => $usu]; // Obtiene el dato del token
    }
    echo json_encode($res, TRUE);
  }
break;
}
?>
```

En la aplicación cliente usamos *cURL* para realizar las solicitudes al servidor, de forma que se ha creado una función genérica para enviar la solicitudes de forma similar a la función que usamos para conectar a las bases de datos. Esta función es (`curl_conexion.php`) :

```
<?php
/*
 * Función de conexión mediante cURL con servidores REST
 * Toma la $URL y el método GET, POST, PUT o DELETE.
 * Opcionalmente parámetros para el paso tipo POST y un token
 * de autenticación JWT
 */
```

```
function curl_conexion($url, $metodo, $token = NULL, $params = NULL){
    $curl = curl_init();
    curl_setopt($curl, CURLOPT_URL, $url);
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, TRUE);
    curl_setopt($curl, CURLOPT_CUSTOMREQUEST, $metodo);
    if($params != NULL){
      curl_setopt($curl,CURLOPT_POSTFIELDS, http_build_query($params));
    }
    if($token != NULL){
      curl_setopt($curl, CURLOPT_HTTPHEADER, array("Authorization:
Bearer " . $token , "cache-control: no-cache"));
    }
    else{
      curl_setopt($curl, CURLOPT_HTTPHEADER, array("cache-control: no-cache"));
    }
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, TRUE);
    // Ejecuta la llamada al servidor y obtiene la respuesta
    $response = curl_exec($curl);
    $err = curl_error($curl);
    curl_close($curl);
    if ($err) {
           $response = json_encode("cURL Error #:" . $err);
    }
    return $response;
}
```

En la instrucción destacada: **curl_setopt($curl, CURLOPT_HTTPHEADER, array("Authorization: Bearer " . $token , "cache-control: no-cache"));** se manda el token en la cabecea **Authorization: Bearer** que es el formato necesario para enviar este elemento.

Para crear el token la aplicación cliente del ejemplo pasa una solicitud con el nombre de usuario y su contraseña al servidor y el token obtenido lo almacena en una *sesión* para poder enviarlo al servidor cada vez que se deba realizar una petición. Se recomienda que se active el protocolo HTTPS para aumentar la seguridad. El token también puede guardarse en una *cookie* o en la propia página en un elemento de formulario *input* tipo *hidden.*

Código de ejemplo

En *Autenticar_JWT_PHP.zip*