

Programación III

API REST - SLIM

Clase 8

Maximiliano Neiner

Temas a Tratar

- ✖ Definición de Middleware (PSR7 PSR15)
- ✖ Middleware en Slim 4

Temas a Tratar

- ✖ Definición de Middleware (PSR7 PSR15)
- ✖ Middleware en Slim 4

Middleware - PSR7 PSR15 (1/3)

- ✖ Middleware es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos.
- ✖ Un middleware implementa la interface PSR15:
 - \Psr\Http\Message\ServerRequestInterface - El objeto de solicitud PSR7 (parámetro).
 - \Psr\Http\Server\RequestHandlerInterface - El objeto controlador de solicitudes PSR15 (parámetro).
 - \Psr\Http\Message\ResponseInterface - El objeto de respuesta PSR7 (retorno).

Middleware - PSR7 PSR15 (2/3)

✖ PHP Standards Recommendations.

✖ Documentación oficial:

- <http://www.php-fig.org/psr/psr-7/>
- <http://www.php-fig.org/psr/psr-15/>

✖ Un mensaje HTTP es una petición de un cliente a un servidor o una respuesta de un servidor a un cliente.

✖ Estas especificaciones definen interfaces para los mensajes HTTP:

- [Psr\Http\Message\RequestInterface](#) (PSR7)
- [Psr\Http\Message\RequestHandlerInterface](#) (PSR15)

Middleware - PSR7 PSR15 (3/3)

- ✖ El único requisito es que un middleware DEBE devolver una instancia de \Psr\Http\Message\ResponseInterface.
- ✖ Cada middleware DEBERÍA invocar al siguiente middleware y pasarle los objetos Request y Response como argumentos.

Temas a Tratar

- ✖ Definición de Middleware (PSR7 PSR15)
- ✖ Middleware en Slim 4

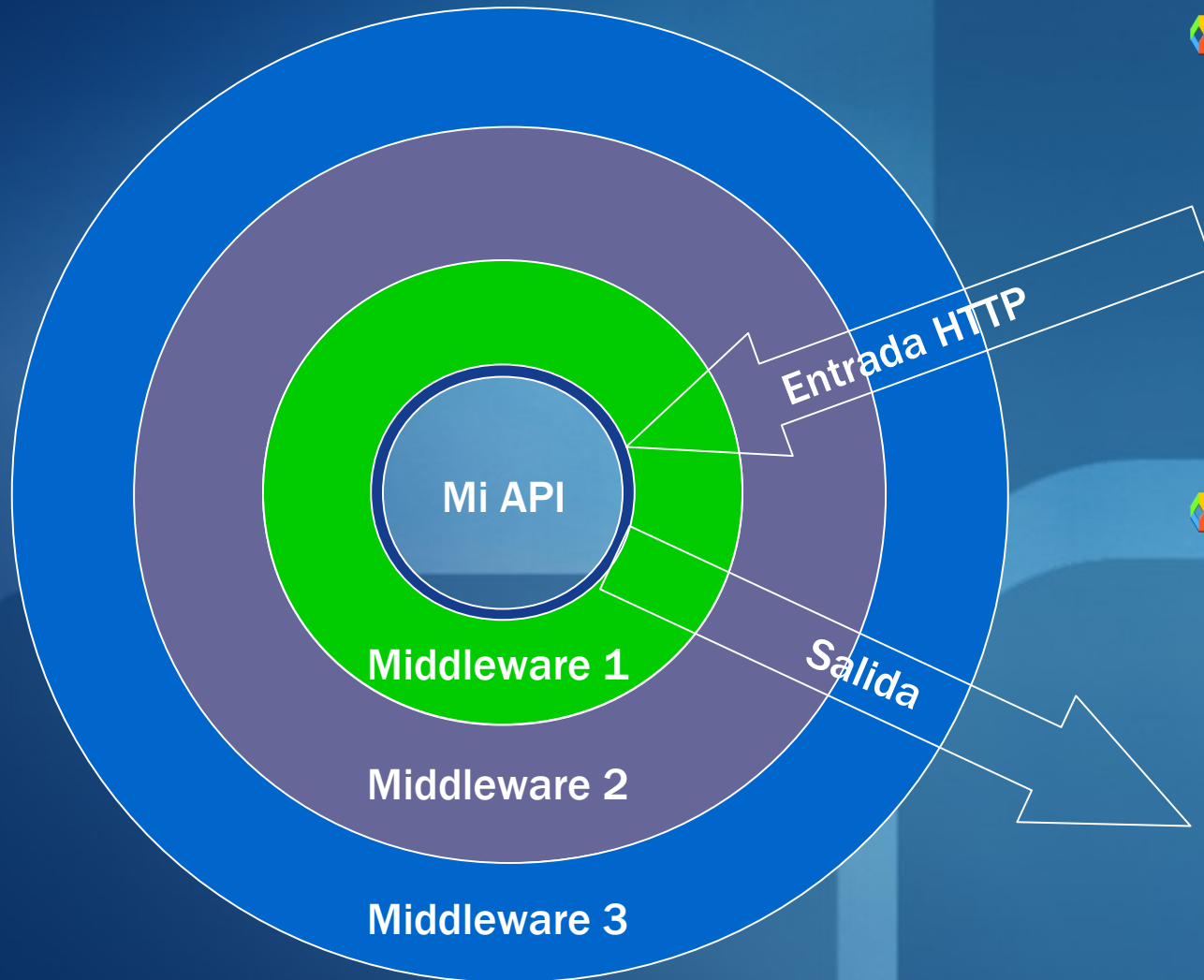
Middleware en Slim 4 (1/3)

- ✖ En Slim, podemos ejecutar código antes y después de una llamada a nuestra APIRest, para poder manipular los objetos Request y Response como mejor nos parezca.
- ✖ Esto se realiza por medio de un middleware.
- ✖ Posibles usos:
 - Proteger la aplicación de la falsificación de solicitudes cruzadas.
 - Autenticar las solicitudes antes de ejecutar su aplicación.
 - Etc., etc., etc.

Middleware en Slim 4 (2/3)

- ✖ Slim añade middleware como capas concéntricas que rodean su aplicación principal.
- ✖ Cada nueva capa de middleware rodea cualquier capa de middleware existente.
- ✖ La estructura concéntrica se expande hacia afuera a medida que se añaden capas de middleware adicionales.
- ✖ La última capa de middleware agregada es la primera en ser ejecutada.

Middleware en Slim 4 (3/3)



Entrada:

- 3
- 2
- 1
- Mi API



Salida

- Mi API
- 1
- 2
- 3

Temas a Tratar

✖ Definición de Middleware (PSR7 PSR15)

✖ Middleware en Slim 4

- Funciones middleware
- Middleware Route
- Middleware Group y Map
- Middleware con P00
- Utilidades

Funciones Middleware

- ✘ Para definir la función se debe respetar la firma de la estandarización PSR15.

```
$mwUno = function (Request $request, RequestHandler $handler) : ResponseMW {  
  
    //EJECUTO ACCIONES ANTES DE INVOCAR AL VERBO  
    $antes = " en MW_UNO antes del callable <br>";  
  
    //INVOCO AL VERBO  
    $response = $handler->handle($request);  
  
    //OBTENGO LA RESPUESTA DEL VERBO  
    $contenidoAPI = (string) $response->getBody();  
  
    //GENERO UNA NUEVA RESPUESTA  
    $response = new ResponseMW();  
  
    //EJECUTO ACCIONES DESPUES DE INVOCAR AL VERBO  
    $despues = " en MW_UNO después del callable <br>";  
  
    $response->getBody()->write("{ $antes} { $contenidoAPI} <br> { $despues}");  
  
    return $response;  
};
```

- ✘ Para agregarla, se hace con el método de instancia add(), de application, route, map o group.

```
$app->add($mwUno);
```

Temas a Tratar

✖ Definición de Middleware (PSR7 PSR15)

✖ **Middleware en Slim 4**

- Funciones middleware
- **Middleware Route**
- Middleware Group y Map
- Middleware con P00
- Utilidades

Middleware - Route (1/2)

- ✖ El 'middleware de ruta' solo se invoca si su ruta coincide con el método de solicitud HTTP actual y la URI.
- ✖ Este middleware es ejecutado inmediatamente después de invocar cualquiera de los métodos de enrutamiento de la aplicación Slim (por ejemplo, get o post).
- ✖ El middleware se agrega con el método add() de la instancia Route.

Middleware - Route (2/2)

```
$app->put('/param/', function (Request $request, Response $response, array $args) : Response {  
  
    $response->getBody()->write("API => PUT");  
    return $response;  
  
})->add(function (Request $request, RequestHandler $handler) : ResponseMW {  
  
    //EJECUTO ACCIONES ANTES DE INVOCAR AL SIGUIENTE MW  
    $antes = " en MW_PUT antes del callable <br>";  
  
    //INVOCO AL SIGUIENTE MW  
    $response = $handler->handle($request);  
  
    //OBTENGO LA RESPUESTA DEL MW  
    $contenidoAPI = (string) $response->getBody();  
  
    //GENERO UNA NUEVA RESPUESTA  
    $response = new ResponseMW();  
  
    //EJECUTO ACCIONES DESPUES DE INVOCAR AL SIGUIENTE MW  
    $despues = " en MW_PUT después del callable ";  
  
    $response->getBody()->write("{ $antes } { $contenidoAPI } <br> { $despues }");  
  
    return $response;  
  
});
```

Temas a Tratar

✖ Definición de Middleware en PSR7 PSR15

✖ **Middleware en Slim 4**

- Funciones middleware
- Middleware Route
- **Middleware Group y Map**
- Middleware con P00
- Utilidades

Middleware - Group (1/2)

- ✖ Los middlewares, además de poder agregarse a la aplicación y a las rutas, pueden también agregarse a los métodos de grupos, como a las rutas individuales internas.
- ✖ El 'middleware de grupo' solo se invoca si su ruta coincide con uno de los métodos de solicitud HTTP definidos y los URI del grupo.
- ✖ Para agregar el middleware dentro de un *callback* individual o para todo el grupo, se establecerá mediante el método de instancia `add()`.

Middleware - Group (2/2)

```
$grupo->get('/hora', function (Request $request, Response $response, array $args) : Response {  
  
    $response->getBody()->write(date('H:i:s'));  
    return $response;  
  
})->add(function(Request $request, RequestHandler $handler) : ResponseMW {  
  
    //EJECUTO ACCIONES ANTES DE INVOCAR AL SIGUIENTE MW  
    $antes = " en MW_GRUPO_DOS antes del callable <br>";  
  
    //INVOCO AL SIGUIENTE MW  
    $response = $handler->handle($request);  
  
    //OBTENGO LA RESPUESTA DEL MW  
    $contenidoAPI = (string) $response->getBody();  
  
    //GENERO UNA NUEVA RESPUESTA  
    $response = new ResponseMW();  
  
    //EJECUTO ACCIONES DESPUES DE INVOCAR AL SIGUIENTE MW  
    $despues = " en MW_GRUPO_DOS después del callable ";  
  
    $response->getBody()->write("{ $antes} { $contenidoAPI} <br> { $despues}");  
  
    return $response;  
});
```

Middleware - Map

```
$app->map(['GET', 'POST'], '/mapeado', function (Request $request, Response $response, array $args) : Response {  
  
    $response->getBody()->write("API => GET o POST");  
    return $response;  
  
})->add(function(Request $request, RequestHandler $handler) : ResponseMW {  
  
    if($request->getMethod() === "GET")  
    {  
        $respuesta = 'Entro por GET';  
    }  
    else if($request->getMethod() === "POST")  
    {  
        $respuesta = 'Entro por POST';  
    }  
  
    //INVOCO AL SIGUIENTE MW  
    $response = $handler->handle($request);  
  
    //OBTENGO LA RESPUESTA DEL MW  
    $contenidoAPI = (string) $response->getBody();  
  
    //GENERO UNA NUEVA RESPUESTA  
    $response = new ResponseMW();  
  
    $response->getBody()->write("{ $respuesta} <br> { $contenidoAPI}");  
  
    return $response;  
  
});
```

EJERCICIOS

Ejercicio (1/4)

AGREGAR EL GRUPO /CREDENCIALES CON LOS VERBOS GET Y POST (MOSTRAR QUE VERBO ES).

AL GRUPO, AGREGARLE UN MW QUE, DE ACUERDO EL VERBO, VERIFIQUE CREDENCIALES O NO.

GET -> NO VERIFICA. ACCEDE AL VERBO.

POST-> VERIFICA; SE ENVIA: NOMBRE Y PERFIL.

*- SI EL PERFIL ES 'ADMINISTRADOR', MUESTRA EL NOMBRE Y ACCEDE AL VERBO.

<input checked="" type="checkbox"/>	nombre	Juan
<input checked="" type="checkbox"/>	perfil	administrador

*- SI NO, MUESTRA MENSAJE DE ERROR.

NO ACCEDE AL VERBO.

nombre	Juan
perfil	empleado

NO necesita credenciales para GET.

API => GET

Vuelvo del verificador de credenciales.

Verifico credenciales:

Bienvenido Juan

API => POST

Vuelvo del verificador de credenciales.

Verifico credenciales:

No tienes habilitado el ingreso.

Vuelvo del verificador de credenciales.

Ejercicio (2/4)

AGREGAR EL GRUPO /JSON CON LOS VERBOS GET Y POST. RETORNA UN JSON (MENSAJE, STATUS).

AL GRUPO, AGREGARLE UN MW QUE, DE ACUERDO EL VERBO, VERIFIQUE CREDENCIALES O NO.

GET -> NO VERIFICA. ACCEDE AL VERBO. RETORNA {"API=>GET", 200}.

```
{  
  "mensaje": "API => GET"  
}
```

POST-> VERIFICA; SE ENVIA (JSON): OBJ_JSON, CON NOMBRE Y PERFIL.

*- SI EL PERFIL ES 'ADMINISTRADOR', ACCEDE AL VERBO. RETORNA {"API=>POST", 200}.

KEY	VALUE
obj_json	{"nombre":"juancito", "perfil":"administrador"}

```
{  
  "mensaje": "API => POST"  
}
```

*- SI NO, MUESTRA MENSAJE DE ERROR. NO ACCEDE AL VERBO. RETORNA {"ERROR. NOMBRE", 403}

obj_json	{"nombre":"juancito", "perfil":"empleado"}
----------	--

```
{  
  "mensaje": "ERROR. juancito sin permisos."  
}
```

Temas a Tratar

✖ Definición de Middleware (PSR7 PSR15)

✖ **Middleware en Slim 4**

- Funciones middleware
- Middleware Route
- Middleware Group y Map
- **Middleware con P00**
- Utilidades

Middleware con P00 (1/2)

✖ Se pueden crear clases con los métodos que puedan ser usados como middleware

```
use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Server\RequestHandlerInterface as RequestHandler;
use Slim\Psr7\Response as ResponseMW;

class MiClase
{
    public function MostrarInstancia(Request $request, RequestHandler $handler) : ResponseMW
    {
        //EJECUTO ACCIONES ANTES DE INVOCAR AL SIGUIENTE MW
        $antes = " Desde método de instancia (antes del verbo)<br>";

        //INVOCO AL SIGUIENTE MW
        $response = $handler->handle($request);

        //OBTENGO LA RESPUESTA DEL MW
        $contenidoAPI = (string) $response->getBody();

        //GENERO UNA NUEVA RESPUESTA
        $response = new ResponseMW();

        //EJECUTO ACCIONES DESPUES DE INVOCAR AL SIGUIENTE MW
        $despues = " Desde método de instancia (después del verbo) ";

        $response->getBody()->write("{ $antes} { $contenidoAPI} <br> { $despues}");

        return $response;
    }
}
```


Middleware con POO (2/2)



Lo agrego a la api, rutas, map o grupos.

```
//VERBOS EN RUTA
$grupo->get('/hoy', function (Request $request, Response $response, array $args) : Response {

    $response->getBody()->write(date('Y-m-d'));
    return $response;
});

//MW A NIVEL DE RUTA
$grupo->get('/hora', function (Request $request, Response $response, array $args) : Response {

    date_default_timezone_set('America/Argentina/Buenos_Aires');
    $response->getBody()->write(date('H:i:s'));
    return $response;

})->add(\MiClase::class . ":MostrarInstancia");

//MW A NIVEL DE MAP
$grupo->map(['PUT', 'DELETE'], '/map', function (Request $request, Response $response, array $args) : Response {

    $response->getBody()->write('API => PUT o DELETE - En verbo dentro de map.');
```

return \$response;

```
})->add(\MiClase::class . "::MostrarEstatico")
->add(\MiClase::class . ":MostrarInstancia");

//MW A NIVEL DE GRUPO
})->add(\MiClase::class . "::MostrarEstatico");
```

EJERCICIOS

Ejercicio (3/4)

AGREGAR EL GRUPO /JSON_BD CON LOS VERBOS GET Y POST (A NIVEL RAIZ).

GET Y POST -> TRAEN (EN FORMATO JSON) TODOS LOS USUARIO DE LA BASE DE DATOS. USUARIO->TRAERTODOS().

```
[
  {
    "id": 1,
    "nombre": "juan",
    "apellido": "perez",
    "correo": "juan@perez.com",
    "foto": "1_20200606100000_perez.jpg",
    "id_perfil": 1,
    "clave": "123456"
  },
  {
    "id": 2,
    "nombre": "pedro",
    "apellido": "garcia",
    "correo": "pedro@garcia.com",
    "foto": "2_20200606100000_pedro.jpg",
    "id_perfil": 2,
    "clave": "654321"
  }
]
```

AGREGAR UN MW, PARA POST, QUE VERIFIQUE AL USUARIO (CORREO Y CLAVE).

POST-> VERIFICADORA->VERIFICARUSUARIO(); SE ENVIA(JSON): OBJ_JSON, CON CORREO Y CLAVE.

KEY	VALUE
obj_json	{"correo":"juan@perez.com", "clave":"123456"}

Ejercicio (3/4)

*- SI EXISTE EL USUARIO EN LA BASE DE DATOS
(VERIFICADORA::EXISTEUSUARIO(\$OBJ)), ACCEDE AL VERBO.

```
[
  {
    "id": 1,
    "nombre": "juan",
    "apellido": "perez",
    "correo": "juan@perez.com",
    "foto": "1_20200606100000_perez.jpg",
    "id_perfil": 1,
    "clave": "123456"
  },
  {
    "id": 2,
    "nombre": "pedro",
    "apellido": "garcia",
    "correo": "pedro@garcia.com",
    "foto": "2_20200606100000_pedro.jpg",
    "id_perfil": 2,
    "clave": "654321"
  }
]
```

*- SI NO, MUESTRA MENSAJE DE ERROR. NO ACCEDE AL VERBO.
{"ERROR.", 403}

KEY	VALUE
obj_json	{"correo":"juan@perez.com", "clave":"123"}

```
{
  "mensaje": "ERROR. Correo o clave incorrectas."
}
```

Ejercicio (4/4)

AGREGAR, A NIVEL DE GRUPO UN MW, QUE VERIFIQUE:

GET-> ACCEDE AL VERBO. (NO HACE NADA NUEVO).

POST-> VERIFICA SI FUE ENVIADO EL PARAMETRO 'OBJ_JSON'.

*- SI NO, MUESTRA MENSAJE DE ERROR. NO ACCEDE AL VERBO.
{ "ERROR.", 403}.

```
{
  "mensaje": "Falta parámetro obj_json!!!"
}
```

*- SI FUE ENVIADO, VERIFICA SI EXISTEN LOS ATRIBUTOS 'CORREO' Y 'CLAVE'.

- SI ALGUNO NO EXISTE (O LOS DOS), MUESTRA MENSAJE DE ERROR. NO ACCEDE AL VERBO. { "ERROR.", 403}.

obj_json	{}
----------	----

```
{
  "mensaje": "Falta atributo correo!!!Falta atributo clave!!!"
}
```

obj_json	{"clave":"123456"}
----------	--------------------

```
{
  "mensaje": "Falta atributo correo!!!"
}
```

obj_json	{"correo":"juan@perez.com"}
----------	-----------------------------

```
{
  "mensaje": "Falta atributo clave!!!"
}
```

*- SI EXISTEN, ACCEDE AL VERBO.

Temas a Tratar

✖ Definición de Middleware (PSR7 PSR15)

✖ **Middleware en Slim 4**

- Funciones middleware
- Middleware Route
- Middleware Group y Map
- Middleware con P00
- **Utilidades**

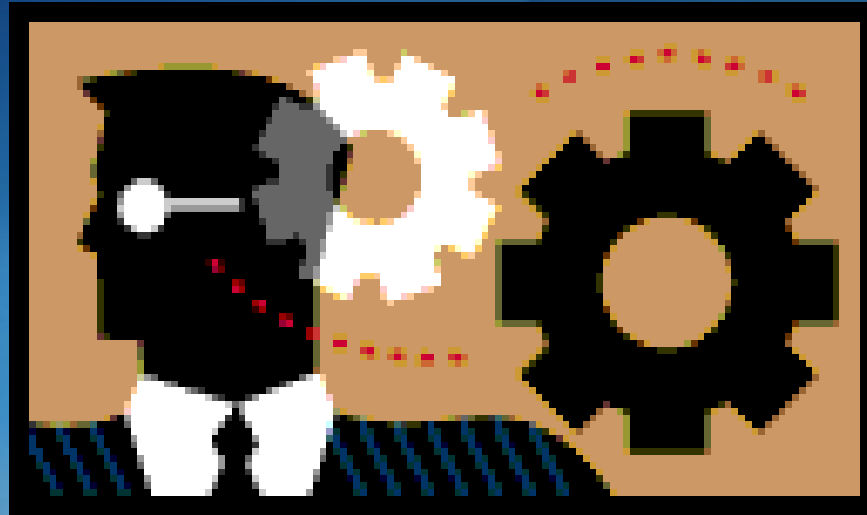
Funciones Middleware



Posibles utilidades:

- Habilitar CORS (Cross Origin Resource Sharing)
- Manipular archivos, modificar tamaño de imágenes.
- Encriptar, firewall, validaciones.
- Verificador de credenciales (JWT)
- Información del cliente: Geolocalización, dispositivos, IPs.
- Querés MÁS??

<https://github.com/middlewares/awesome-psr15-middlewares>



Ejercitación

Ejercicios

- ✖ Hacer un middleware de aplicación que tome usuario y contraseña y verifique en BD.
- ✖ Hacer middleware de grupo, solo para post, que permita agregar un nuevo usuario, sólo si el perfil es 'admin'.
- ✖ Hacer middleware de grupo, solo para delete, que permita borrar un usuario, si el perfil es 'super_admin'.
- ✖ Hacer middleware de ruta, solo para put y get, que tome el tiempo de demora entre que entra y sale la petición.