UCH UNIVERSIDAD DE CIENCIAS Y HUMANIDADES

DESARROLLO WEB

2017-I

SEMANA 11

ANGULARJS – INTEGRACIÓN CON PHP

Eric Gustavo Coronel Castillo

ecoronel@uch.edu.pe - gcoronelc.blogspot.com

ANGULARJS – INTEGRACION CON PHP

El objetivo de esta semana es que integres AngularJS con PHP utilizando JSON.

ÍNDICE

JSON EN PHP 4

PHP: JSON_ENCODE() 4

EJEMPLO 1 4

EJEMPLO 2 5

EJEMPLO 3 6

EJEMPLO 4 7

PROCESAR PARAMETROS JSON 9

PHP: JSON_DECODE() 9

EJEMPLO 5 9

Ejecución 1: Pasar un arreglo JSON de números 10

Ejecución 2: Pasar un objeto JSON 11

Ejecución 3: Pasar un objeto JSON 12

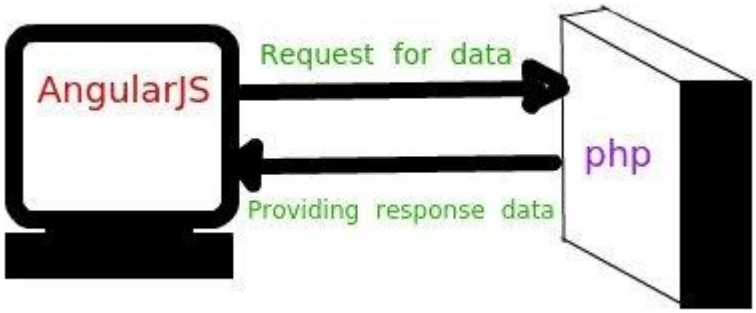
EJEMPLO 6 13

Ejecución 1: Pasar un arreglo JSON de números 14

Ejecución 2: Pasar un objeto JSON 15

Ejecución 3: Pasar un objeto JSON 16

JSON EN PHP



Una de las ventajas que tiene PHP, es que tiene integrada funciones para convertir arreglo y objetos a formato JSON y viceversa, haciendo de esta manera su integración con AngularJS muy fácil.

PHP: json_encode()

Esta función retorna la representación JSON del dato que se le pasa como parámetro.

Ejemplo 1

En este ejemplo se ilustra cómo pasa un arreglo simple a un arreglo JSON.

```
<?php

$datos = array(20, 30, 15, 50);

var_dump($datos);

$json = json_encode($datos);

var_dump($json);

?>
```

El resultado es el siguiente:

```
array (size=4)
  0 => int 20
  1 => int 30
  2 => int 15
  3 => int 50

string '[20,30,15,50]' (length=13)
```

Ejemplo 2

En este ejemplo se aprecia cómo se para un arreglo asociativo a formato JSON.

```
<?php

$datos = array(
 'codigo'=>"123",
 'nombre'=>"TV",
 'precio'=>4500.00,
 'stock'=>500);

var_dump($datos);

$json = json_encode($datos);

var_dump($json);

?>
```

El resultado es el siguiente:

```
array (size=4)
  'codigo' => string '123' (length=3)
  'nombre' => string 'TV' (length=2)
  'precio' => float 4500
  'stock' => int 500

string '{"codigo":"123","nombre":"TV","precio":4500,"stock":500}' (length=56)
```

Ejemplo 3

En este ejemplo se ilustra como pasar un objeto a formato JSON.

```
<?php

$datos = (object) array(
 'codigo'=>"123",
 'nombre'=>"TV",
 'precio'=>4500.00,
 'stock'=>500);

var_dump($datos);

$json = json_encode($datos);

var_dump($json);

?>
```

El resultado es el siguiente:

```
object(stdClass)[1]
  public 'codigo' => string '123' (length=3)
  public 'nombre' => string 'TV' (length=2)
  public 'precio' => float 4500
  public 'stock' => int 500

string '{"codigo":"123","nombre":"TV","precio":4500,"stock":500}' (length=56)
```

Ejemplo 4

En este ejemplo se tiene un arreglo a arreglos asociativos, y se puede apreciar cómo se convierte a formato JSON.

```
<?php

$datos = array(
  array('codigo'=>"P01",'nombre'=>"Gustavo Coronel"),
  array('codigo'=>"P02",'nombre'=>"Ricardo Marcelo"),
  array('codigo'=>"P03",'nombre'=>"Claudia Ramirez"),
  array('codigo'=>"P04",'nombre'=>"Alejandra Ramos")
);

var_dump($datos);

$json = json_encode($datos);

var_dump($json);

?>
```

El resultado es el siguiente:

```
array (size=4)
  0 =>
    array (size=2)
      'codigo' => string 'P01' (length=3)
      'nombre' => string 'Gustavo Coronel' (length=15)
  1 =>
    array (size=2)
      'codigo' => string 'P02' (length=3)
      'nombre' => string 'Ricardo Marcelo' (length=15)
  2 =>
    array (size=2)
      'codigo' => string 'P03' (length=3)
      'nombre' => string 'Claudia Ramirez' (length=15)
  3 =>
    array (size=2)
      'codigo' => string 'P04' (length=3)
      'nombre' => string 'Alejandra Ramos' (length=15)

string '[{"codigo":"P01","nombre":"Gustavo Coronel"},
        {"codigo":"P02","nombre":"Ricardo Marcelo"},
        {"codigo":"P03","nombre":"Claudia Ramirez"},
        {"codigo":"P04","nombre":"Alejandra Ramos"}
       ]' (length=177)
```

PROCESAR PARAMETROS JSON

PHP: json_decode()

Convierte una cadena codificado en JSON a su equivalente en PHP.

Ejemplo 5

En este ejemplo se tiene un programa en PHP que recibe un dato JSON y lo convierte a un elemento PHP.

```
<?php

$json = $_GET["json"];

echo "<h2>CADENA RECIBIDA</h2>";
var_dump($json);

$datos = json_decode($json);

echo "<h2>EQUIVALENTE EN PHP</h2>";
var_dump($datos);

?>
```

Ejecución 1: Pasar un arreglo JSON de números

La llamada podría ser así:

```
demo05.php?json=[100,400,432,567,234]5}
```

El resultado es el siguiente:

CADENA RECIBIDA

```
string '[100,400,432,567,234]' (length=21)
```

EQUIVALENTE EN PHP

```
array (size=5)
  0 => int 100
  1 => int 400
  2 => int 432
  3 => int 567
  4 => int 234
```

En este caso un arreglo JSON se convierte en un arreglo PHP.

Ejecución 2: Pasar un objeto JSON

La llamada podría ser así:

```
demo05.php?json={"n1":"15","n2":"18","n3":"16","n4":"18"}
```

El resultado es el siguiente:

CADENA RECIBIDA

```
string '{"n1":"15","n2":"18","n3":"16","n4":"18"}' (length=41)
```

EQUIVALENTE EN PHP

```
object(stdClass)[1]
  public 'n1' => string '15' (length=2)
  public 'n2' => string '18' (length=2)
  public 'n3' => string '16' (length=2)
  public 'n4' => string '18' (length=2)
```

En este caso, por defecto la función **json_decode()** lo convierte en un objeto PHP.

Ejecución 3: Pasar un objeto JSON

Esta ejecución es similar a la anterior, la diferencia está en que los datos son numéricos:

```
demo05.php?json={"n1":15,"n2":18,"n3":16,"n4":18}
```

El resultado es el siguiente:

CADENA RECIBIDA

```
string '{"n1":15,"n2":18,"n3":16,"n4":18}' (length=33)
```

EQUIVALENTE EN PHP

```
object(stdClass)[1]
  public 'n1' => int 15
  public 'n2' => int 18
  public 'n3' => int 16
  public 'n4' => int 18
```

En este caso, la función **json_decode()** lo convierte en un objeto PHP, pero los campos son numéricos.

Ejemplo 6

En este ejemplo se tiene un programa en PHP que recibe un dato JSON y lo convierte a arreglo PHP.

```
<?php

$json = $_GET["json"];

echo "<h2>CADENA RECIBIDA</h2>";
var_dump($json);

$datos = json_decode($json);

echo "<h2>EQUIVALENTE EN PHP</h2>";
var_dump($datos);

?>
```

Ejecución 1: Pasar un arreglo JSON de números

La llamada podría ser así:

```
demo06.php?json=[100,400,432,567,234]5}
```

El resultado es el siguiente:

CADENA RECIBIDA

```
string '[100,400,432,567,234]' (length=21)
```

EQUIVALENTE EN PHP

```
array (size=5)
  0 => int 100
  1 => int 400
  2 => int 432
  3 => int 567
  4 => int 234
```

En este caso un arreglo JSON se convierte en un arreglo PHP.

Ejecución 2: Pasar un objeto JSON

La llamada podría ser así:

```
demo06.php?json={"n1":"15","n2":"18","n3":"16","n4":"18"}
```

El resultado es el siguiente:

CADENA RECIBIDA

```
string '{"n1":"15","n2":"18","n3":"16","n4":"18"}' (length=41)
```

EQUIVALENTE EN PHP

```
array (size=4)
  'n1' => string '15' (length=2)
  'n2' => string '18' (length=2)
  'n3' => string '16' (length=2)
  'n4' => string '18' (length=2)
```

En este caso, la función **json_decode()** está convirtiendo un objeto JSON en un arreglo asociativo de PHP.

Ejecución 3: Pasar un objeto JSON

Esta ejecución es similar a la anterior, la diferencia está en que los datos son numéricos:

```
demo06.php?json={"n1":15,"n2":18,"n3":16,"n4":18}
```

El resultado es el siguiente:

CADENA RECIBIDA

```
string '{"n1":15,"n2":18,"n3":16,"n4":18}' (length=33)
```

EQUIVALENTE EN PHP

```
array (size=4)
  'n1' => int 15
  'n2' => int 18
  'n3' => int 16
  'n4' => int 18
```

En este caso, la función **json_decode()** está convirtiendo un objeto JSON en un arreglo asociativo de PHP.