Clase 15

Análisis de Sistemas

Materia:
Programación Web II

Docente contenidista: ROLDÁN, Hernán

Revisión: Coordinación

Contenido

Clase 15

¡Te damos la bienvenida a la materia **Programación Web II**!

En esta clase vamos a ver los siguientes temas:

- Introducción: qué son las expresiones regulares.
- Sintaxis básica y patrones de búsqueda.
- Clases de caracteres y negaciones.
- Algunos ejemplos de expresiones regulares.

PHP Expresiones regulares
Introducción:
Qué son las expresiones regulares

Las expresiones regulares, también conocidas como regex o regexp, son secuencias de caracteres que definen un patrón de búsqueda en un texto.

Son una poderosa herramienta utilizada en programación para manipular, buscar, validar y reemplazar cadenas de caracteres de forma flexible y precisa.

Su finalidad es proporcionar un método eficiente y versátil para trabajar con cadenas de texto, permitiendo encontrar patrones específicos, realizar operaciones complejas y extraer información relevante.

Al utilizar expresiones regulares, podemos resolver una amplia variedad de tareas, tales como:

- **Búsqueda de patrones:** Podemos buscar cadenas de texto que cumplan con un patrón específico, como palabras clave, números de teléfono, direcciones de correo electrónico, etc.
- **Validación de datos:** Las expresiones regulares son muy útiles para validar datos ingresados por usuarios, como correos electrónicos, números de teléfono, códigos postales, etc. Podemos asegurarnos de que los datos cumplan con un formato adecuado antes de procesarlos.
- **Reemplazo de texto:** Podemos reemplazar una o todas las ocurrencias de un patrón con otro texto. Esto es útil para normalizar datos o realizar cambios masivos en un texto.
- **Extracción de información:** Podemos extraer información específica de una cadena de texto, como obtener todos los números que aparecen en un texto o extraer partes específicas de una URL.
- **Análisis de texto estructurado**: Las expresiones regulares son útiles para analizar y manipular texto estructurado, como archivos de configuración o registros de datos con un formato específico.

En resumen, las expresiones regulares son una herramienta esencial para trabajar con texto de manera flexible y eficiente.

Permiten resolver una amplia gama de tareas relacionadas con el manejo de cadenas, desde búsquedas y validaciones hasta reemplazos y extracciones de información.

Al dominar el uso de las expresiones regulares, podemos mejorar la eficacia y precisión en el procesamiento y manipulación de datos en nuestras aplicaciones.

Sintaxis básica y patrones de búsqueda

Las expresiones regulares se definen utilizando una sintaxis especial que permite describir patrones de búsqueda. Cada carácter en una expresión regular representa un carácter que se debe buscar o una instrucción especial que define un patrón más complejo.

A continuación, se presentan algunos conceptos básicos de la sintaxis de expresiones regulares:

- **Caracteres Literales:** Los caracteres literales son aquellos que se buscan exactamente como están escritos en la expresión regular. Por ejemplo, la expresión regular **/abc/** buscará la cadena **"abc"** en el texto.
- **Metacaracteres:** Los metacaracteres son caracteres especiales que tienen un significado especial en las expresiones regulares. Algunos de los metacaracteres más comunes son: **. (punto)** que representa cualquier carácter, *** (asterisco)** que indica cero o más repeticiones del carácter anterior, **+ (más)** que indica una o más repeticiones del carácter anterior, **? (signo de interrogación)** que indica cero o una repetición del carácter anterior, entre otros.
- **Clases de Caracteres:** Las clases de caracteres se definen entre **corchetes [ ]** y permiten buscar cualquier carácter que se encuentre dentro de la clase. Por ejemplo, la expresión regular **/[aeiou]/** buscará cualquier vocal en el texto.

- **Negación de Clases de Caracteres:** Se puede utilizar el carácter **^** dentro de una clase de caracteres para indicar la negación, es decir, buscar cualquier carácter que no esté dentro de la clase. Por ejemplo, la expresión **regular /[^0-9]/** buscará cualquier carácter que no sea un dígito numérico.
- **Anclajes:** Los anclajes son metacaracteres que indican el inicio o fin de una línea o cadena de texto. Por ejemplo, el metacaracter **^** al comienzo de la expresión regular indica que la búsqueda debe comenzar desde el inicio de la línea, mientras que el metacaracter **$** al final de la expresión regular indica que la búsqueda debe finalizar en el fin de la línea.

Patrones de Búsqueda

Un patrón de búsqueda es simplemente una secuencia de caracteres y metacaracteres que conforman una expresión regular. Los patrones definen qué cadenas de texto queremos encontrar en el texto que estamos buscando.

Por ejemplo, si tenemos el siguiente texto:

```
El lenguaje PHP es muy popular
```

Y queremos encontrar la palabra "PHP", el patrón de búsqueda sería simplemente PHP. Si queremos encontrar todas las vocales en el texto, el patrón sería [aeiou].

Es importante destacar que los patrones de búsqueda pueden ser tan simples o complejos como lo requiera la situación. Con la combinación adecuada de caracteres y metacaracteres, se pueden definir patrones precisos y eficientes para encontrar la información deseada en un texto.

En conclusión, la sintaxis básica de las expresiones regulares se compone de caracteres literales y metacaracteres que permiten definir patrones de búsqueda. Estos patrones son fundamentales para encontrar y manipular cadenas de texto de manera efectiva en programación. A medida que profundices en el uso de las expresiones regulares, podrás desarrollar patrones más complejos y utilizar herramientas poderosas para resolver diversos problemas relacionados con el manejo de texto.

Caracteres especiales

Los caracteres especiales en expresiones regulares son aquellos que tienen un significado especial y no se interpretan como caracteres literales.

En cambio, se utilizan para definir patrones más complejos de búsqueda.

Algunos de los caracteres especiales más comunes son:

- **. (Punto):** Representa cualquier carácter, excepto un salto de línea. Por ejemplo, la expresión regular a.b coincidirá con "aab", "acb", "adb", etc., pero no con "a\nb".
- **^ (Circunflejo):** Indica el inicio de una línea o de una cadena de texto. Por ejemplo, la expresión regular ^abc coincidirá con "abc" cuando aparezca al principio de una línea o de la cadena.
- **$ (Dólar):** Indica el final de una línea o de una cadena de texto. Por ejemplo, la expresión regular xyz$ coincidirá con "xyz" cuando aparezca al final de una línea o de la cadena.
- **| (Barra vertical):** Indica una alternativa entre dos patrones. Por ejemplo, la expresión regular a|b coincidirá con "a" o "b".
- **[] (Corchetes):** Se utiliza para definir una clase de caracteres. Por ejemplo, la expresión regular [aeiou] coincidirá con cualquier vocal en el texto.
- **[^] (Corchetes negados):** Se utiliza para definir una clase de caracteres negada. Por ejemplo, la expresión regular [^0-9] coincidirá con cualquier carácter que no sea un dígito numérico.
- **() (Paréntesis):** Se utilizan para agrupar partes de un patrón y aplicar cuantificadores o realizar extracción mediante grupos de captura.

Cuantificadores

Los cuantificadores son metacaracteres que se aplican a un elemento o grupo de elementos en una expresión regular y definen la cantidad de veces que ese elemento o grupo debe aparecer en el texto.

Algunos de los cuantificadores más comunes son:

- **(Asterisco):** Indica que el elemento o grupo anterior puede aparecer cero o más veces. Por ejemplo, la expresión regular a* coincidirá con "a", "aa", "aaa", etc.
- **+ (Más):** Indica que el elemento o grupo anterior debe aparecer una o más veces. Por ejemplo, la expresión regular a+ coincidirá con "a", "aa", "aaa", pero no con la cadena vacía.
- **? (Signo de interrogación):** Indica que el elemento o grupo anterior puede aparecer cero o una vez. Por ejemplo, la expresión regular a? coincidirá con "a" o la cadena vacía.
- **{n} (Llaves):** Indica que el elemento o grupo anterior debe aparecer exactamente "n" veces. Por ejemplo, la expresión regular a{3} coincidirá con "aaa".
- **{n, m} (Llaves con dos valores):** Indica que el elemento o grupo anterior debe aparecer al menos "n" veces y como máximo "m" veces. Por ejemplo, la expresión regular a{2,4} coincidirá con "aa", "aaa" o "aaaa".

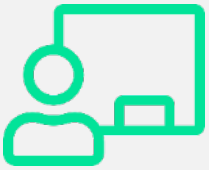
Los caracteres especiales y cuantificadores son herramientas poderosas que permiten definir patrones de búsqueda flexibles y precisos.

Con su uso adecuado, podremos realizar operaciones más complejas y eficientes en el procesamiento y manipulación de cadenas de texto en tus aplicaciones PHP.

Funciones de expresiones regulares

Las funciones de expresiones regulares en PHP son un conjunto de funciones incorporadas que permiten trabajar con expresiones regulares y realizar diversas operaciones relacionadas con el manejo de cadenas de texto.

Estas funciones son proporcionadas por PHP para facilitar la manipulación y análisis de texto utilizando patrones de búsqueda definidos mediante expresiones regulares.



A continuación, explicaremos las funciones de expresiones regulares más comunes en PHP:

- **preg_match():**
  Esta función se utiliza para realizar una búsqueda de un patrón en una cadena de texto. Devuelve 1 si el patrón se encuentra al menos una vez en la cadena, o 0 si no se encuentra. También se puede utilizar para extraer partes de una cadena que coincidan con el patrón mediante el uso de grupos de captura.
- **preg_match_all():**
  Similar a preg_match(), pero en lugar de detenerse al encontrar la primera coincidencia, busca todas las ocurrencias del patrón en la cadena y las devuelve como un array.
- **preg_replace():**
  Esta función se utiliza para buscar un patrón en una cadena y reemplazarlo por otro texto. Es útil para realizar reemplazos masivos o normalizar datos en una cadena.
- **preg_split():**
  Permite dividir una cadena en función de un patrón de búsqueda. El resultado es un array de subcadenas obtenidas al dividir la cadena original según el patrón especificado.

Estas funciones toman como primer argumento la expresión regular que define el patrón de búsqueda y como segundo argumento la cadena de texto en la que se realizará la búsqueda o manipulación. Además, algunas de estas funciones admiten argumentos adicionales para ajustar el comportamiento de la búsqueda, como los modificadores de expresiones regulares (p. ej. i, m, s, x) para hacer

que la búsqueda sea insensible a mayúsculas y minúsculas, que tenga en cuenta saltos de línea, o que permita comentarios en la expresión regular, entre otros.

Recordá que: las funciones de expresiones regulares en PHP son herramientas poderosas y versátiles para trabajar con cadenas de texto y resolver problemas relacionados con el manejo de datos.

Dominar su uso te permitirá realizar operaciones más complejas y eficientes en tus aplicaciones, como validaciones de datos, extracción de información o filtrado de texto. Sin embargo, es importante tener en cuenta que el manejo de expresiones regulares puede volverse complejo con patrones más avanzados, por lo que se recomienda estudiar y practicar para mejorar tus habilidades en el uso de estas funciones

Continuando con la explicación, aquí están algunas funciones adicionales relacionadas con expresiones regulares en PHP:

- **preg_grep():** Esta función se utiliza para buscar elementos en un array que coincidan con un patrón de expresión regular. Devuelve un nuevo array que contiene sólo los elementos que coinciden con el patrón.
- **preg_quote():** Esta función se utiliza para escapar los caracteres especiales en una cadena, de modo que puedan ser interpretados literalmente en una expresión regular. Es útil cuando se quiere buscar una cadena literal en lugar de un patrón de búsqueda.
- **preg_filter():** Similar a preg_replace(), pero en lugar de realizar el reemplazo, devuelve una copia de la cadena original con las coincidencias reemplazadas. La cadena original no se modifica.
- **preg_last_error():** Esta función devuelve el código del último error ocurrido al evaluar una expresión regular. Puede ser útil para depurar problemas en la ejecución de las expresiones regulares.

Clases de caracteres y negaciones

En expresiones regulares, las "Clases de caracteres" y las "Negaciones" son conceptos relacionados que permiten buscar un conjunto específico de caracteres dentro de una cadena de texto.

Clase de Caracteres

Una "Clase de caracteres" se define utilizando corchetes [ ] y permite especificar un conjunto de caracteres entre los cuales se buscará una coincidencia.

Por ejemplo, la expresión regular [aeiou] buscará cualquier vocal en el texto, ya que las letras "a", "e", "i", "o" y "u" están incluidas en la clase.

Ejemplo:

- **Expresión regular:** [aeiou]
- **Cadena de texto:** "Hola, este es un ejemplo."
- **Coincidencias:** "o", "a", "e", "e", "e", "u"

En este ejemplo, la expresión regular busca cualquier vocal en la cadena de texto y encuentra todas las ocurrencias de "o", "a", "e" y "u".

Negación de Clases de Caracteres

Cuando se utiliza el carácter ^ dentro de una clase de caracteres, se indica una "Negación". Esto significa que la expresión regular buscará cualquier carácter que NO esté incluido dentro de la clase.

Ejemplo:

- **Expresión regular:** [^0-9]
- **Cadena de texto:** "Hola, este es un ejemplo 123."
- **Coincidencias:** "H", "o", "l", "a", ",", " ", "e", "s", "t", "e", " ", "e", "s", " ", "u", "n", " ", "e", "j", "e", "m", "p", "l", "o", " ", "."

En este ejemplo, la expresión regular busca cualquier carácter que NO sea un dígito numérico (0 al 9) en la cadena de texto, y encuentra todas las letras, espacios y caracteres especiales que no son dígitos.

Las clases de caracteres y las negaciones son muy útiles para definir patrones de búsqueda más flexibles y específicos en expresiones regulares.

Pueden ser combinadas con otros metacaracteres y cuantificadores para realizar búsquedas más complejas en textos y realizar manipulaciones de datos más precisas en aplicaciones PHP.

Es importante tener en cuenta que la posición de los caracteres dentro de la clase de caracteres no afecta a la búsqueda, ya que la expresión regular buscará cualquier coincidencia de los caracteres especificados, independientemente del orden en que aparezcan.

Algunos ejemplos de expresiones regulares

Ejemplo de validación de correos electrónicos:

```php
<?php

  $email = "usuario@example.com";

  if(preg_match("/^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/",
$email)){
    echo "El correo electrónico es válido.";
  }
  else{
      echo "El correo electrónico no es válido.";
  }

?>
```

En este ejemplo, utilizamos la función **preg_match()** para validar un correo electrónico. La expresión regular **/^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/** busca patrones que coincidan con el formato típico de un correo electrónico.

Si el correo es válido, muestra un mensaje indicando que es válido; de lo contrario, muestra un mensaje de error.

Expresión regular explicada

^: Este símbolo indica que la coincidencia debe ocurrir al inicio de la cadena. Es decir, el correo electrónico válido debe comenzar con el patrón definido a continuación.

[a-zA-Z0-9._%+-]+: Esta parte del patrón define el nombre de usuario del correo electrónico. Los corchetes [] indican una clase de caracteres, y dentro de ella se especifican los caracteres permitidos para el nombre de usuario: letras mayúsculas y minúsculas (a-z y A-Z), dígitos numéricos (0-9) y algunos caracteres especiales permitidos en los correos electrónicos como . (punto), _ (guión bajo), %, + y -. El símbolo + indica que el nombre de usuario debe tener uno o más caracteres.

@: Es un carácter literal que indica la presencia del símbolo "@" en el correo electrónico.

[a-zA-Z0-9.-]+: Esta parte del patrón define el dominio del correo electrónico. Nuevamente, se utiliza una clase de caracteres para especificar los caracteres permitidos: letras mayúsculas y minúsculas (a-z y A-Z), dígitos numéricos (0-9) y los caracteres . (punto) y - (guión). El símbolo + indica que el dominio debe tener uno o más caracteres.

\.: Este es un carácter literal que representa un punto. Es necesario escapar el punto con una barra invertida (\) porque en expresiones regulares el punto tiene un significado especial y queremos buscar un punto literal.

[a-zA-Z]{2,}: Esta parte del patrón define la extensión del dominio, que debe tener al menos dos caracteres. Nuevamente, se utilizan las letras mayúsculas y minúsculas (a-z y A-Z) en una clase de caracteres.

$: Este símbolo indica que la coincidencia debe ocurrir al final de la cadena. Es decir, el correo electrónico válido debe terminar con el patrón definido anteriormente.

Ejemplo de extracción de información:

```php
<?php

 $texto = "Hola, mi número de teléfono es 123-456-7890 y mi correo es usuario@example.com.";

 preg_match("/\d{3}-\d{3}-\d{4}/", $texto, $matches);

 echo "Número de teléfono encontrado: " . $matches[0];

?>
```

Expresión regular explicada:

En este ejemplo, utilizamos preg_match() para buscar un número de teléfono en el texto. La expresión regular /\d{3}-\d{3}-\d{4}/ busca un patrón con tres dígitos, un guión, otros tres dígitos, otro guión y cuatro dígitos. Si encuentra el número de teléfono en el texto, lo muestra en pantalla.

Ejemplo de reemplazo de palabras:

```php
<?php

  // Ejemplo 3: expresión regular para reemplazar palabras

  $texto = "El lenguaje PHP es muy popular.";

  $nuevoTexto = preg_replace("/PHP/", "JavaScript", $texto);

  echo "Texto original: " . $texto;
  echo "\n";
  echo "Texto modificado: " . $nuevoTexto;

?>
```

En este ejemplo, utilizamos preg_replace() para reemplazar una palabra en el texto. La expresión regular "/PHP/" busca la palabra "PHP" y la reemplaza con "JavaScript". El resultado muestra el texto original y el texto modificado con el reemplazo realizado.

***En conclusión**:*

El uso de expresiones regulares es de suma importancia en programación, especialmente en PHP, debido a su capacidad para realizar búsquedas, validaciones y manipulaciones complejas en cadenas de texto de manera eficiente y precisa.

Al utilizar expresiones regulares, podemos resolver diversos problemas relacionados con la manipulación y verificación de datos, como validar correos electrónicos, números de teléfono, formatos de fechas, entre otros.

Hemos llegado así al final de esta clase en la que vimos:

- Introducción: qué son las expresiones regulares.
- Sintaxis básica y patrones de búsqueda.
- Clases de caracteres y negaciones.
- Algunos ejemplos de expresiones regulares.

Te esperamos en la **clase en vivo** de esta semana.
No olvides realizar el **desafío semanal**.

¡Hasta la próxima clase!

Bibliografía

Documentación Oficial de PHP

W3Schools: PHP Tutorial

Cabezas Granado, Luis. González Lozano, F. (2018). Desarrollo web con PHP y MySQL. Anaya Multimedia.

Heurtel, O. (2016) Desarrolle un sitio web dinámico e interactivo. Eni Ediciones.

Welling, Luke Thompson, L. (2017) Desarrollo Web con PHP y MySQL. Quinta Edición (2017, Editorial Anaya).

Para ampliar la información

PHP: Funciones de PCRE

PHP: Regular Expression Functions

PHP: The Right Way

Todo sobre PHP

PHP Programming Language Tutorial - Full Course

PHP Full Course for non-haters

CURSO de php desde cero

MDN: Introducción al lado Servidor

Guía de HTML