

Desarrollo de Aplicaciones Web

Expresiones Regulares

Expresiones Regulares.

Durante el proceso de intercambio de datos con el servidor se han de intercambiar datos consistentes, para no corromper los datos almacenados en las bases de datos asociadas a aquel.

Solución → Previo al intercambio, habrá que filtrar los datos para:

- **Analizar** que los campos obligatorios están cubiertos.
- **Comprobar** que el tipo de información solicitada es correcta.

Expresiones Regulares.

Análisis de campos obligatorios.- desarrollando funciones del tipo:

- isEmpty() → ¿está vacío? → usar *valor != null , valor.length != 0*
- isNumber() → ¿es un dígito? → usar función *isNaN()* *JavaScript*
- isInteger() → ¿es un entero?
- isPosInteger() → ¿es un entero positivo?

Expresiones Regulares.

Comprobación del tipo de información:

Expresiones Regulares → permiten definir el patrón de una cadena de caracteres, siendo de especial interés para la creación de formularios y efectos varios.

Tipos de patrones:

- Cuantificadores
- Caracteres de escape
- Clases o conjuntos de caracteres
- Caracteres de marcado o posición
- Alternancias
- Agrupaciones
- Referencias inversas
- Patrones misceláneos

Expresiones Regulares.

Mecanismos de definición:

- Mediante barras inclinadas

Ej.- /patrón/opciones.

- Mediante el constructor de la clase RegExp

Ej.- var exp_reg
exp_reg= new **RegExp**("patron", ["opciones"])

Opciones:

- i.- búsqueda haciendo caso omiso de la distinción entre mayúsculas y minúsculas.
- g.- búsqueda global (en toda la cadena/hasta la primera coincidencia)

Expresiones Regulares.

Patrones Validos:

- **Cuantificadores**.- permiten agregar datos sobre cantidades de caracteres a las expresiones regulares

* → cero o más veces

+ → una o más veces

? → cero o una vez

Expresiones Regulares.

Patrones Validos:

- **Caracteres de escape.**- sirven para localizar caracteres no imprimibles y para representar caracteres con significado especial en la expresión.

\ → carácter de escape

Expresiones Regulares.

Patrones Validos:

- **Clases o conjuntos de caracteres.**- sirven para localizar cualquiera de los caracteres que forman parte del conjunto .

[conjunto de caracteres] → caracteres que pueden formar parte de la cadena

\b conjunto de caracteres \b → sólo esos caracteres tal como aparecen en el conjunto

\B conjunto de caracteres \B → niega *\b*

Ej.- [A-Za-z]

Expresiones Regulares.

Patrones Validos:

- **Caracteres de marcado de posición.**- marcan una posición específica dentro de la cadena.

$\wedge \rightarrow$ comienzo de cadena o línea

$\$ \rightarrow$ final de cadena o línea

Expresiones Regulares.

Patrones Validos:

- **Alternancias**.- permiten definir opciones para una expresión

$$a \mid b \rightarrow a \text{ ó } b$$

Expresiones Regulares.

Patrones Validos:

- **Agrupaciones**.- permiten capturar subpatrones, dentro de un patrón

(*patrón*) → la coincidencia se recupera con
\$0 ... \$9 o con \1 ... \9

Ej.- (A-Za-z)\1

Expresiones Regulares.

Patrones Validos:

Otros patrones

- $\{n\} \rightarrow$ coincide exactamente n veces.
- $\{n, \}$ \rightarrow coincide como mínimo n veces.
- $\{n, m\} \rightarrow$ coincide n veces mínimo y m veces máximo.

Ej.- $[a-z] \{2, \}$

Expresiones Regulares.

Patrones Validos:

Otros patrones

- $\wedge \rightarrow$ niega lo que sigue
- $\cdot \rightarrow$ cualquier carácter excepto retorno de carro.
- $\backslash d \rightarrow [0-9]$.
- $\backslash w \rightarrow [A-Za-z0-9_]$.
- $\backslash W \rightarrow$ niega $\backslash w$.

Expresiones Regulares.

Patrones Validos:

Otros patrones

- `\n` → coincide con carácter nueva línea.
- `\r` → coincide con retorno de carro.
- `\t` → coincide con tabulador.
- `\f` → coincide con avance de página.
- `\v` → coincide con tabulador vertical.
- `\s` → coincide con `[\n\r\t\f\v]`

Expresiones Regulares.

Patrones Validos:

Otros patrones

- $?=$ → búsqueda anticipada positiva $Ej \rightarrow q(?=u|a)$
- $?!$ → búsqueda anticipada negativa $Ej \rightarrow q(!=u|a)$
- $?:$ → subexpresión que coincide con modelo, pero no captura coincidencia $Ej \rightarrow cas(?:a|as)$

Expresiones Regulares.

Precedencia entre operadores:

- \
- (), (? :), (? !), []
- *, +, ?, {n}, {n,}, {n, m}
- ^, \$
- |

Expresiones Regulares.

El método `test`.- método de RegExp que permite analizar la existencia de coincidencias entre un patrón y un string de caracteres.

Ejemplo: dirección de e-mail válida.

```
Function isEmail(sEMail)
{
    var re= /^[a-z0-9\-\.\.]+@[a-z0-9\-\.\.]+\.[a-z]{2,}$/i

    if (re.test(SEMail))
        return true;
    else
        return false;
}
```

Expresiones Regulares.

El método `exec`.- método de RegExp que permite obtener un array con los elementos individualizados de una cadena de caracteres a partir de una expresión regular.

Ejemplo: partes de una dirección de e-mail.

```
var re= /^[a-z0-9\-\.\.]+@([a-z0-9\-\.\.]+\.[a-z]{2,})$/i
```

```
var email= re.exec(SEMail);
```

Almacena:

email[0].- dirección completa.

email[1].- usuario.

email[2].- dominio.

Nota:

- RegExp.\$1= email[1]
- RegExp.\$2= email[2]

Expresiones Regulares. HTML5.

Atributo **pattern**.- permite que a la etiqueta `<input>` le sea insertado un patrón válido en formato de expresión regular.

Ej.-

Numero: `<input type="number" name="number" pattern="[0-9]{1,3}">`
`<input type="submit" value="enviar">`

Numero:

Por favor, ajústese al formato solicitado.