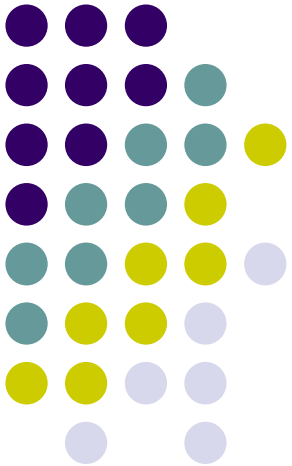
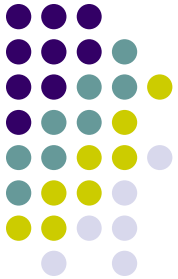
Validación Form





Some of the information you entered is missing or incorrect. Please check all highlighted messages below.

- Please enter Last Name using letters, apostrophes or dashes.
- Enter a valid date for Date of Birth.
- Please enter a valid e-mail address.

Personal Info

First Name: Marty Stepp

Last Name:

Date of Birth: January Day Year

E-mail Address: foo@bar

Identify yourself by your:

- Account Number
- ATM/Debit Card
- Credit Card

Para que necesitamos validar un form?

- **validación**: consiste en asegurar que los valores de un form sean correctos
- Algunos ejemplos de validación pueden ser:
  - Evitar los espacios en blanco (email)
  - Asegurar que los tipos de valores sean los correctos
    - Enteros, tipos monedas, números de teléfonos, DNI, email, fechas, tarjeta de crédito, ...
  - Asegurar que los valores sean coherentes (que los dos email entrados correspondan)
  - Seguridad, evitar que código incorrecto (inyección de código) se efectúe

Peligros de códigos enviado en forms HTML

- inyección de código
- Un usuario podría enviar en un form código no deseado

<h1>pwned</h1>

  - Una solución seria restringir el uso de “<“ y “>”
  - Pero htmlspecialchars devuelve una versión de un string que puede ser igualmente interpretada

```
$text = "<p>hi 2 u & me</p>";
$text = htmlspecialchars($text);   # "&lt;p&gt;hi 2 u &amp; me&lt;/p&gt;“
```

validación cliente vs. server

- La validación puede ser efectuada:
- Lado cliente (antes que el form sea enviado)
  - Puede permitir una mejor usabilidad por el usuario, pero no es segura (porque?)
- Lado servidor (con código PHP, después que el form se ha enviado)
  - Necesaria para una validación realmente segura, pero mas lenta
- Ambas
  - Mejor balance entre usabilidad y seguridad, pero necesita mas esfuerzo de programación.

Ejemplo de validación lado cliente

Con HTML

```
<form action="http://prueba.com/prueba.php" method="get">
    <div>
        City:  <input name="city" /> <br />
        State: <input name="state" size="2" maxlength="2" /> <br />
        ZIP:   <input name="zip" size="5" maxlength="5" /> <br />
        <input type="submit" />
    </div>
</form>
```

Como se obtendría el mismo resultado en el servidor con PHP?

Solución?

1) Podemos escribir código para evitar que el usuario teclee los caracteres no deseados
2) o podemos permitirlo, pero no ejecutarlos

- Ejemplo:

```
$city  = $_POST["city"];
$state = $_POST["state"];
$zip   = $_POST["zip"];
if (!$city || strlen($state) != 2 || strlen($zip) != 5) {
    print "Error, invalid city/state/zip submitted.";
}
```

- *Idea básica:* examinar los valores de los parámetros, y si no están en el rango enseñar un error pero:
  - Como ver la diferencia entre integer, reales, y strings?
  - Como testear un numero de tarjeta de crédito valido?
  - Como testear si una persona tiene mas que un apellido o nombres?
  - (o como testear si un string tiene un formato complejo?)

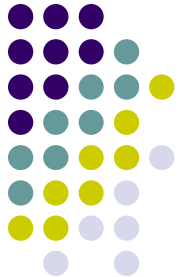
Regex

- /^[a-zA-Z_\-]+@(([a-zA-Z_\-])+\.)+[a-zA-Z]{2,4}$/
- Les expresiones regulares ("regex"): son una descripción de un patrón de texto
- Permiten testear si un string se parece a un patrón de texto
- Ejemplo /abc/
- en PHP, los “regex” son string que empiezan y terminan por /
- El “regex” /abc/ puede corresponder a cualquier string que contenga “abc”, como

Si: "abc", "abcdef", "defabc", ".=.abc.=.", ...

NO: "fedcba", "ab c", "PHP", ...

Comodín



- Un punto . Corresponde a cualquier carácter excepto el \n “line break”.
  - /.oo.y/ matches "Doocy", "goofy", "LooNy", ...
- Un carácter i (de cola) al final de un regex (después de cerrar /) significa una correspondencia case-insensitive
  - /mart/i corresponde a "Marty Stepp", "smart fellow", "WALMART", ...

Caracteres especiales

- | significa *OR*
  - /abc|def|g/ corresponde a "abc", "def", o "g"
- () son para agrupamientos
  - /(Homer|Marge) Simpson/ corresponde a "Homer Simpson" o "Marge Simpson"
- \ empieza una secuencia especial
  - Muchos caracteres necesitan este símbolo para que correspondan exactamente: / \ $. [ ] ( ) ^ * + ?
    - /<br \/>/ corresponde al la etiqueta

Cuantificadores

- * significa 0 o mas ocurrencias
  - /abc*/ corresponde a "ab", "abc", "abcc", "abccc", ...
  - /a(bc)*/ corresponde "a", "abc", "abcbc", "abcbcbc", ...
  - /a.*a/ corresponde "aa", "aba", "a8qa", "a!?xyz__9a", ...
- + significa 1 o mas ocurrencias
  - /a(bc)+/ corresponde "abc", "abcbc", "abcbcbc", ...
  - /Goo+gle/ corresponde "Google", "Gooogle", "Goooogle",
- ? significa 0 o 1 ocurrencias
  - /a(bc)?/ corresponde "a" or "abc"

Otros cuantificadores

- {*min*,*max*} significa un número de ocurrencias entre *min* y *max* (incluidos)
  - /a(bc){2,4}/ corresponde a "abcbc", "abcbcbc", or "abcbcbcbc"
- *min* o *max* pueden ser omitidos para especifica cualquier numero
  - {2,} significa 2 o mas
  - {,6} significa hasta a 6
  - {3} means exactamente 3

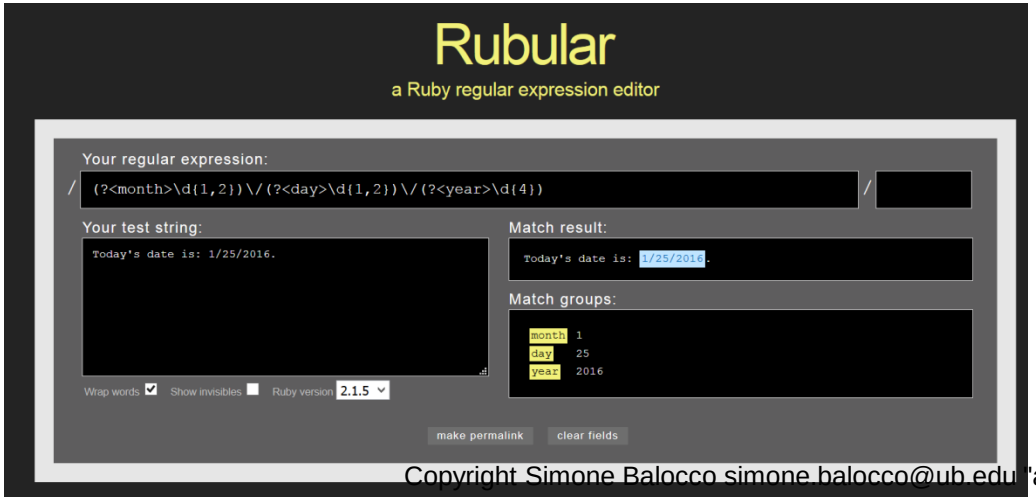
Respuesta

- Escribir un regex que devuelva positivo en el caso de string que contengan:

"Google", "Gooogle", "Goooogle“

sol:

- Se puede usar el sitio web Rubular para testear las expresiones regulares



Anclaje

- ^ representa el inicio de una string o una línea;
  $ representa el final
  - /Jess/ corresponde a todas las string que contienen Jess;
    /^Jess/ corresponde a todas que empiezan con Jess;
    /Jess$/ corresponde a todas que acaban con Jess;
    /^Jess$/ corresponde exactamente a "Jess" only
  - /^Mart.*Stepp$/ corresponde "MartStepp", "Marty Stepp", "Martin D Stepp", ...
    Pero NO "Marty Stepp stinks" o "I H8 Martin Stepp"

Set de caracteres

- [] grupo de caracteres incluido en un set de caracteres; corresponderá a cualquier carácter del set singularmente
  - /[bcd]art/ corresponde a "bart", "cart", and "dart"
  - Equivalente a /(b|c|d)art/ pero mas corto
- Al interior de [], los caracteres de especiales actúan como caracteres normales
  - /what[!*?]*/ matches "what", "what!", "what?**!", "what??!", ...

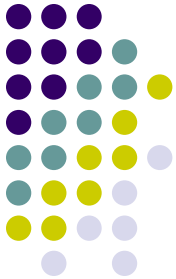
Rango de caracteres [inicio-fin]

- Al interior de un set de caracteres, especifica su rango
  - /[a-z]/ corresponde a cualquier letra minúscula
  - /[a-zA-Z0-9]/ corresponde a cualquier letra o numero minúsculo o mayúsculo
- Un caracteres inicial ^ al interior de un set lo niega.
  - /[^abcd]/ corresponde a cualquier carácter diferente de “a, b, c, or d”
- Al interior del un set de caracteres, - debe comenzar con \ para funcionar
  - /[+\-]?[0-9]+/ corresponde a + o -, seguido por lo menos un número de una cifra
- Ejemplo para definir las notas de asignaturas A, B+, o D- como se escribiría?
  - /[ABCDF][+\-]?/

Secuencias especiales

- Secuencias especiales de un set de caracteres :
  - \d corresponde a cualquier numero (parecido a [0-9]);
  - \D a cualquier non-numero ([^0-9])
  - \w corresponde a cualquier carácter (same as [a-zA-Z_0-9]);
  - \W a cualquier non carácter
  - \s corresponde a cualquier espacio blanco (, \t, \n, etc.);
  - \S a cualquier espacio non blanco
- Cual expresión “regex” corresponde a una cantidad de por lo menos $100.00 ?
  - /\$\d{3,}\.\d{2}/

Ejercicios colaborativos



- https://codeshare.io/COW
- Si quiero comprobar las notas universitarias del sistema anglófono ej: A, B+, or D- ? (test)
- Si quiero comprobar los números de estudiantes de los alumnos que tienen 7 dígitos? (test)
- Cual expresión “regex” corresponde a una cantidad de por lo menos $100.00 ?

Usar “regex” en PHP

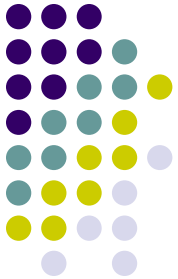
función

- preg_match(*regex*, *string*)
- preg_replace(*regex*, *replacement*, *string*)
- preg_split(*regex*, *string*)

descripción

- devuelve TRUE si *string* corresponde a *regex*
- devuelve un nuevo string con todos los substring que contienen *regex* en lugar de *replacement*
- Devuelve un array de string a partir del *string* de entrada separado por el *regex* indicado

Ejemplo de uso



- Controlar el input del usuario

```
$state = $_POST["state"];
if (!preg_match("/^[A-Z]{2}$/", $state)) {
    print "Error, invalid state submitted.";
}
```

- Cambiar vocales con estrellas

```
$str = "the quick    brown       fox";
$str = preg_replace("/[aeiou]/", "*", $str);
                    #
```

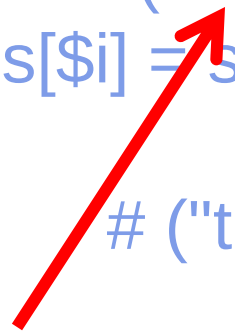
- Separar palabras

```
$str = "th* q**ck br*wn f*x“
$words = preg_split("/[  ]/", $str);
                    # ("th*", "q**ck", "br*wn", "f*x")
```

Ejemplo de uso 1

- Transforma en mayúsculo las palabras que contienen mas que dos vocales consecutivas

```
$words= "th*", "q**ck", "br*wn", "f*x"
for ($i = 0; $i < count($words); $i++) {
    if (preg_match("/\*{2,}/", $words[$i])) {
        $words[$i] = strtoupper($words[$i]);
    }
}                    # ("th*", "Q**CK", "br*wn", "f*x")
```



- * es un simbulo especial que significa 0 o mas ocurrencias
  - Pero aquí queremos identificar el carácter
  - Por esta razon necesitamos usar decir que “\ empieza una secuencia especial”

Para redirigir a otras paginas

- La función header puede ser usada para diferentes mensajes HTTP
  - Enviar códigos de error (404 not found, 403 forbidden, etc.)

```
header("HTTP header text");
```

  - Redirigir de una pagina a otra (en este caso tiene que aparecer antes cualquier otro comando PHP

```
header("Location: url");
```

Ejemplo redireccion en caso de fallo regex

```
$city  = $_POST["city"];
$state = $_POST["state"];
$zip   = $_POST["zip"];
if (!$city || strlen($state) != 2 || strlen($zip) != 5) {
  header("Location: start-page.php");   # invalid input; redirect
}
```

- El problema es que el usuario es re-dirigido a la pagina inicial sin saber que ha pasado.

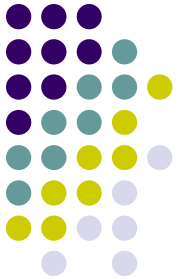
La funcion de stop “die”

- PHP contiene una función que visualiza un mensaje y luego para la ejecución del código

```
die(“mensaje de error");
```

- El problema es la mala experiencia del usuario que solo recibe una pagina de error

Código de ejemplo



```
function check_valid($regex, $param) {
  if (preg_match($regex, $_POST[$param])) {
    return $_POST[$param];
  } else {

    die("Bad $param");
  }
}
```

excepciones

- Se pueden usar excepciones para redirigir el usuario en otras paginas en el caso que ocurra un error especifico.
- Cuando una excepción se activa:
  - La ejecución del código se interrumpe y se graba (y se congela)
  - El código redirige en un “handler” predeterminado
  - el “handler” puede:
    - Re-toma la ejecución del código desde el punto de interrupción
    - Interrumpe de forma definitiva el script
    - Continua el script en otro lugar del código

Crear una exception

```php
<?php       //create function with an exception
function checkNum($number) {
  if($number>1) {
    throw new Exception("Value must be 1 or below");
  }
  return true;
}
 //trigger exception
checkNum(2);
?>
```

Genera el siguiente código de error:

Fatal error: Uncaught exception 'Exception'
with message 'Value must be 1 or below' in C:\webfolder\test.php:6
Stack trace: #0 C:\webfolder\test.php(12):
checkNum(28) #1 {main} thrown in **C:\webfolder\test.php** on line **6**

Recuperar el error

```
//trigger exception in a "try" block
try {
  checkNum(2);
  //If the exception is thrown, this text will not be shown
  echo 'If you see this, the number is 1 or below';
}

//catch exception
catch(Exception $e) {
  echo 'Message: ' .$e->getMessage();
}
?>
```

- el error generado es el siguiente:

Message: Value must be 1 or below