PROGRAMMAZIONE WEB

PHP APPROFONDIMENTI

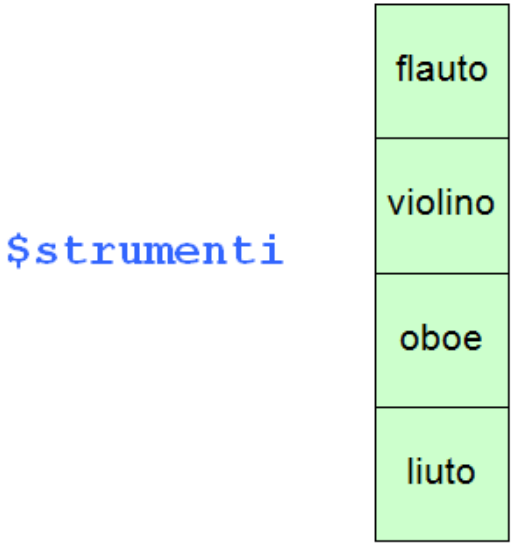
ARRAY E STRINGHE

Prof. Ada Bagozi
ada.bagozi@unibs.it

Array

Array = variabile che contiene un insieme (indicizzato) di **elementi**

- ✓ Elemento: scalare (semplice) o un altro array
- ✓ Individuazione degli elementi mediante indicizzazione → **numerica**, **associativa**



Array indicizzati numericamente

Creazione di array mediante enumerazione dei suoi elementi:

```
$strumenti = array('flauto', 'violino', 'oboe', 'liuto');
```

Creazione di array mediante assegnamento con un altro array:

```
$a = $strumenti;
```

```
$numeri = range(1, 10);
```

⟺

```
$numeri = array(1,2,3,4,5,6,7,8,9,10);
```

```
$lettere = range('a', 'z');
```

⟺

```
$lettere = array('a','b', ..., 'z');
```

Creazione di array mediante caricamento da database

Array associativi

Indice rappresentato da un nome (**chiave**):

```
$prezzi = array('flauto'=>2500,
                'violino'=>8000,
                'oboe'=>5400,
                'liuto'=>11000);
```

flauto	2500
violino	8000
oboe	5400
liuto	11000

Operatory per array

Simbolo	Nome	Esempio	Risultato
+	Unione	$a + $b	Concatenazione di $a con gli elementi di $b che hanno indici diversi da quelli in $a
==	Uguaglianza	$a == $b	Uguaglianza di $a e $b
===	Identità	$a === $b	Identità di $a e $b
!=	Disuguaglianza	$a != $b	Disguaglianza di $a e $b
!==	Non identità	$a !== $b	Non identità di $a e $b

```
$a = array(1,2,3);
$b = array(4,5,6,7,8);
$c = array('1','2','3');
```

⇨

```
$a + $b: (1,2,3,7,8)
$a == $b: false
$a == $c: true
$a === $c: false
```

Array multidimensionali (I)

Indicizzazione numerica:

```
$articoli = array(array('FLT','flauto', 2500),
                  array('VLN', 'violino', 8000),
                  array('OBO', 'oboe', 5400),
                  array('LUT', 'liuto',11000));
```

FLT	flauto	2500
VLN	violino	8000
OBO	oboe	5400
LUT	liuto	11000

Array multidimensionali (II)

Accesso mediante ciclo **for**:

```
for($riga=0; $riga<4; $riga++)
{
  for($colonna=0; $colonna<3; $colonna+
    echo $articoli[$riga][$colonna] . '
  echo '<br/>';
}
```

$riga

$colonna

FLT	flauto	2500
VLN	violino	8000
OBO	oboe	5400
LUT	liuto	11000

⇩

```
FLT flauto 2500
VLN violino 8000
OBO oboe 5400
LUT liuto 11000
```

Array multidimensionali (III)

Indicizzazione mista:

```
$articoli = array(array('codice'=>'FLT',
                        'strumento'=>'flauto',
                        'prezzo'=>2500),
                  array('codice'=>'VLN',
                        'strumento'=> 'violino',
                        'prezzo'=>8000),
                  array('codice'=>'OBO',
                        'strumento'=>'oboe',
                        'prezzo'=>5400),
                  array('codice'=>'LUT',
                        'strumento'=>'liuto',
                        'prezzo'=> 11000));
```

codice	strumento	prezzo
FLT	flauto	2500
VLN	violino	8000
OBO	oboe	5400
LUT	liuto	11000

Array multidimensionali (IV)

Accesso mediante ciclo **foreach**:

```
foreach($articoli as $articolo)
{
  foreach($articolo as $indice => $contenuto)
    echo "$indice: $contenuto ";
  echo '<br/>';
}
```

```
codice: FLT strumento: flauto prezzo: 2500
codice: VLN strumento: violino prezzo: 8000
codice: OBO strumento: oboe prezzo: 5400
codice: LUT strumento: liuto prezzo: 11000
```

$articoli

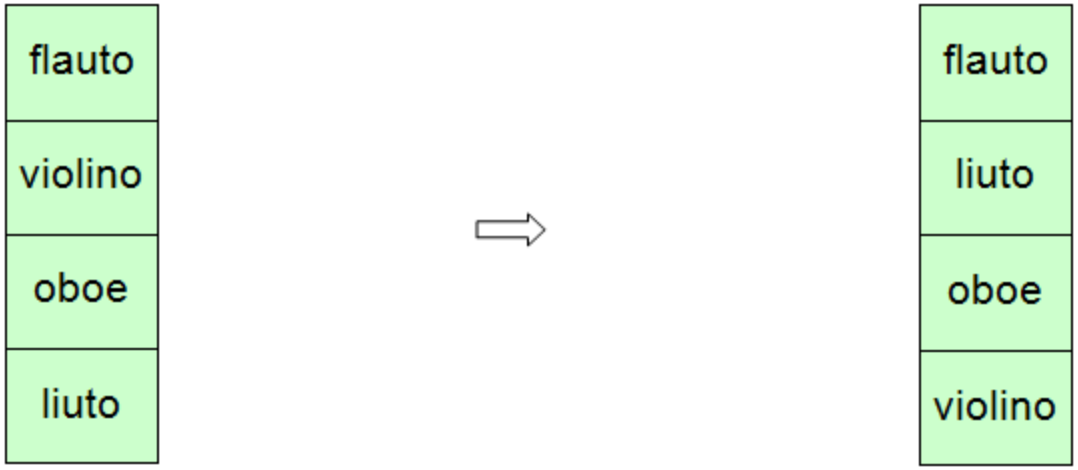
codice	strumento	prezzo
FLT	flauto	2500
VLN	violino	8000
OBO	oboe	5400
LUT	liuto	11000

$articolo

Ordinamento di un array (I)

Ordinamento di array indicizzati numericamente: **`sort()`**

```
$strumenti = array('flauto', 'violino', 'oboe', 'liuto');
sort($strumenti);
```



Ordinamento di un array (II)

Ordinamento di array indicizzati associativamente: **ksort()** e **asort()**

```
$prezzi = array('flauto'=>2500,
                'violino'=>8000,
                'oboe'=>5400,
                'liuto'=>11000);
```

flauto	2500
violino	8000
oboe	5400
liuto	11000

```
ksort($prezzi);
```

flauto	2500
liuto	11000
oboe	5400
violino	8000

```
asort($prezzi);
```

flauto	2500
oboe	5400
violino	8000
liuto	11000

Ordinamento di un array (III)

Ordinamento di array in ordine inverso: **`rsort()`**, **`krsort()`** e **`arsort()`**

```
rsort($strumenti);
```

flauto
violino
oboe
liuto

⇨

violino
oboe
liuto
flauto

```
krsort($prezzi);
```

flauto	2500
violino	8000
oboe	5400
liuto	11000

⇨

violino	8000
oboe	5400
liuto	11000
flauto	2500

```
arsort($prezzi);
```

flauto	2500
violino	8000
oboe	5400
liuto	11000

⇨

liuto	11000
violino	8000
oboe	5400
flauto	2500

Ordinamento casuale di un array


```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Prova</title>
  </head>
  <body>
  <?php
    $a = array(1,2,3);
    $b = array(4,5,6,7,8);
    $c = $a + $b;

    shuffle($c);

    for($i=0; $i<count($c); $i++)
    {
      echo $c[$i] . "<br>";
    }
  ?>
  </body>
</html>
```

Inversione di un array

`array_reverse($a)`: genera un array inverso di **`$a`**

```
$inv_str = array_reverse($strumenti);
```

flauto
violino
oboe
liuto

⇨

liuto
oboe
violino
flauto

```
$inv_art = array_reverse($articoli);
```

codice	strumento	prezzo
FLT	flauto	2500
VLN	violino	8000
OBO	oboe	5400
LUT	liuto	11000

⇨

codice	strumento	prezzo
LUT	liuto	11000
OBO	oboe	5400
VLN	violino	8000
FLT	flauto	2500

Conteggio degli elementi di un array

`count($a)`: restituisce il numero di elementi dell'array `$a`

`array_count_values($a)`: restituisce un array associativo che indica la frequenza di ogni elemento nell'array `$a` (gli elementi di `$a` devono essere scalari)

flauto
violino
flauto
flauto
violino
oboe
liuto
liuto

⇨

flauto	3
violino	2
oboe	1
liuto	2

Da array associativi a variabili scalari

`extract(array a [, int tipologia [, string prefisso]])`: genera un insieme di variabili omonime delle chiavi di ***a***

```
extract($prezzi);
echo "$flauto $violino $oboe $liuto"
```

flauto	2500
violino	8000
oboe	5400
liuto	11000

```
2500 8000 5400 11000
```

Parametri opzionali:

- ✓ ***tipologia***: tipologia di gestione delle collisioni (default: sovrascrittura)
- ✓ ***prefisso***: rilevante per certe tipologie di collisione

tipologia	Significato
EXTR_OVERWRITE	Sovrascrive le variabili in caso di collisione
EXTR_SKIP	Salta l'elemento che provoca la collisione
EXTR_PREFIX_SAME	Crea una variabile `$prefisso_chiave` solo in caso di collisione
EXTR_PREFIX_ALL	Crea tutte le variabili `$prefisso_chiave`, indipendentemente dalle collisioni
EXTR_PREFIX_INVALID	Crea una variabile `$prefisso_chiave` in caso di identificatore (chiave) non valido
EXTR_IF_EXISTS	Estrae solo le variabili che già esistono

Navigazione di un array

Array: ha un puntatore interno che punta all'elemento corrente dell'array (creazione array → punta al primo elemento)

- ✓ `current($a)` → restituisce elemento corrente
- ✓ `each($a)` → restituisce elemento corrente; incrementa puntatore
- ✓ `next($a)` → incrementa puntatore; restituisce elemento corrente
- ✓ `reset($a)` → posiziona puntatore al primo elemento (e lo restituisce)
- ✓ `end($a)` → posiziona puntatore all'ultimo elemento (e lo restituisce)
- ✓ `prev($a)` → decrementa puntatore; restituisce elemento corrente

```
$a = array("alfa", "beta", "gamma");
$stringa = end($a);
while($stringa)
{
  echo "$stringa <br/>";
  $stringa = prev($a);
}
```

⇨

```
gamma
beta
alfa
```

Pulitura di stringhe

`string trim(string stringa)`: rimozione di spaziatura intorno a *stringa*

- Spazio bianco ‘ ‘
- Newline "\n"
- Carriage return "\r"
- Tab orizzontale "\t"
- Tab verticale "\x0B"
- Fine stringa "\0"

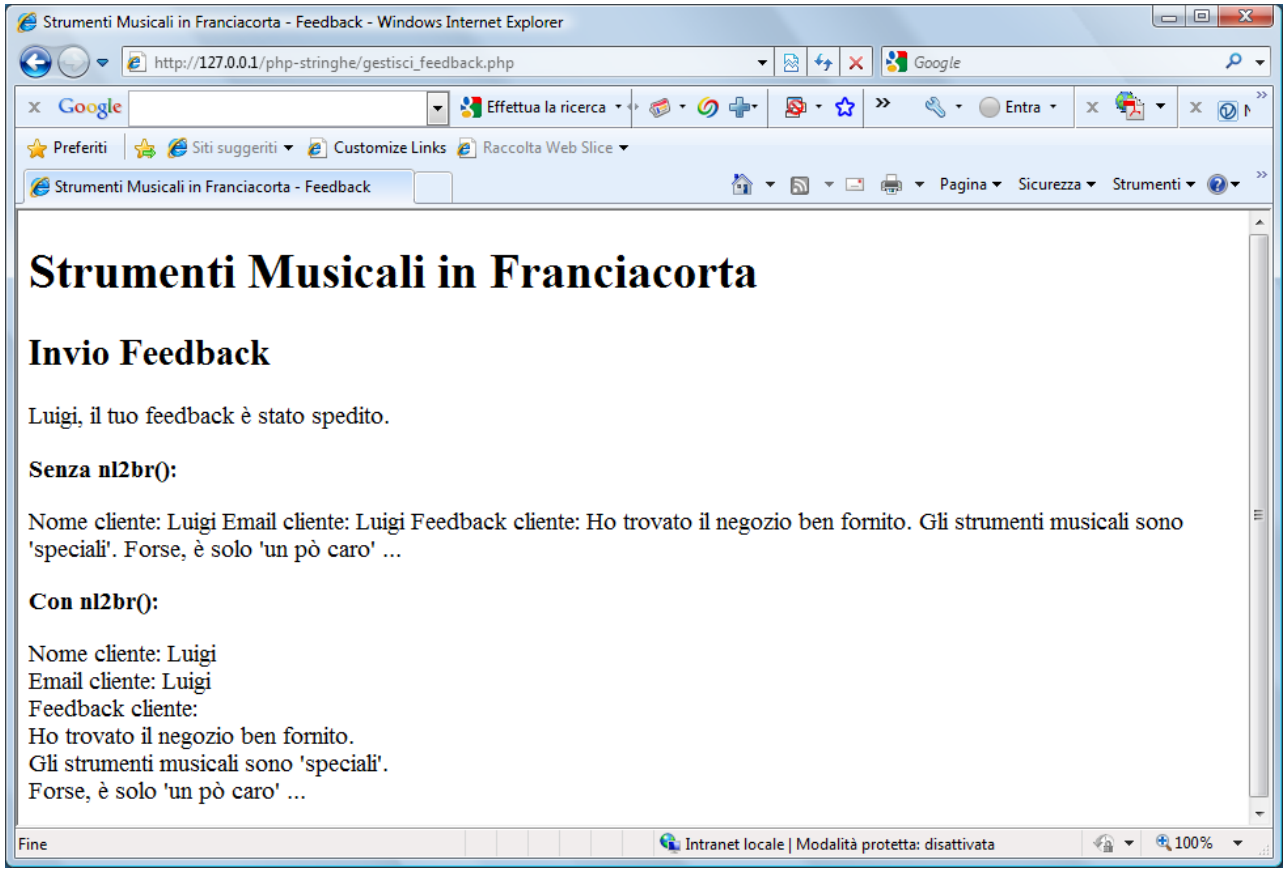
```
$nome = trim($_POST['nome']);
$email = trim($_POST['email']);
```

`string ltrim(string stringa)`: pulitura solo a sinistra

`string rtrim(string stringa)`: pulitura solo a destra

Rendering dei Newline

`string nl2br(string stringa)`: sostituzione di "\n" con "
"



Formattazione di stringhe

void printf(string *formato* [, mixed *args* ...])
string sprintf(string *formato* [, mixed *args* ...])

formato: specifica del formato della stringa di output mediante codici
args: variabili che istanziano i codici nel ***formato***

```
echo "Il costo totale è $costo_totale";
```

⇕

Il costo totale è 83,8

```
printf("Il costo totale è %s", $costo_totale)";
```

```
printf("Il costo totale è %.2f", $costo_totale)";
```

Il costo totale è 83,80

Specifiche di conversione multiple:

```
printf("Il costo totale è %.2f (con spedizione %.2f)",
        $costo_totale, $costo_con_spedizione);
```

Il costo totale è 83.80 (con spedizione 88.80)

Formattazione di stringhe

Sintassi specifica di conversione:

`%['`*carattere-riempimento*] [`-`] [*larghezza*] [`.`*precisione*] *codice*

codice	Interpretazione	Stampa
`b`	Integer	Numero binario
`c`	Integer	Carattere
`d`	Integer	Numero decimale
`f`	Double	Numero in virgola mobile
`o`	Integer	Numero ottale
`s`	String	Stringa
`u`	Integer	Intero senza segno
`x`	Intero	Numero esadecimale con caratteri minuscoli (`a-f`)
`X`	Intero	Numero esadecimale con caratteri maiuscoli (`A-F`)

Formattazione di stringhe

Cambiamento del *case* (maiuscola-minuscola) di stringhe:

```
$subject = 'Feedback dal sito web';
```

Funzione	Effetto	Output
strtoupper($subject)	Uppercase	FEEDBACK DAL SITO WEB
strtolower($subject)	Lowerrcase	feedback dal sito web
ucfirst($subject)	Maiuscola solo del primo carattere della stringa se alfabetico	Feedback dal sito web
ucwords($subject)	Maiuscola del primo carattere di ogni parola della stringa se alfabetico	Feedback Dal Sito Web

Utile nel confronto tra stringhe:

```
if(strtolower($email) == 'luigi.rossi@alice.it')
  $to = 'marketing@strumentifranciacorta.com';
else
  $to = 'info@strumentifranciacorta.com';
```

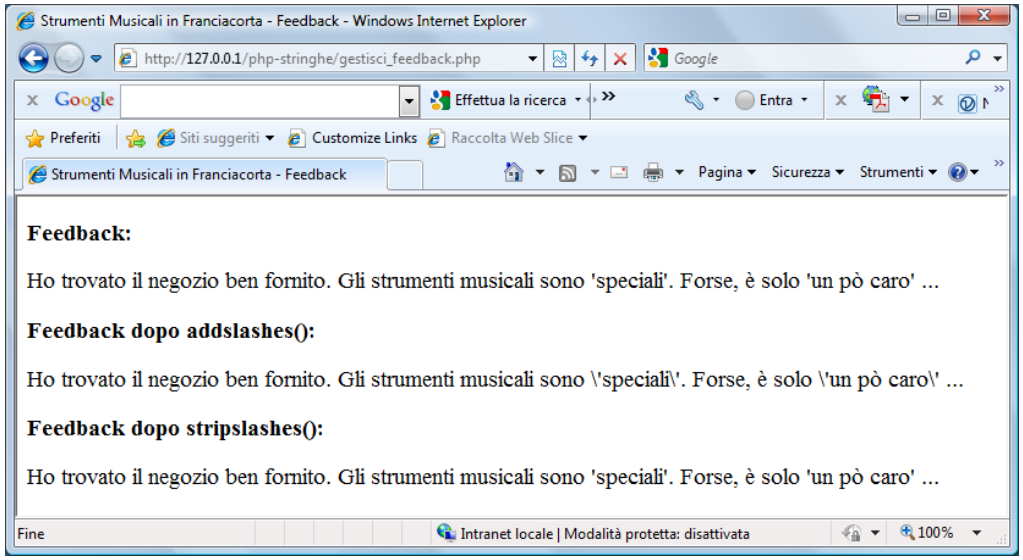
Formattazione di stringhe

Problemi nella memorizzazione di stringhe nel database: apici, \, NULL
Inibizione dei metacaratteri mediante backslash: \', \", \\

`string addslashes(string stringa)`

`string stripslashes(string stringa)`

```
$feedback = addslashes(trim($_POST['feedback']));
$feedback = stripslashes($feedback);
```



Esplosione ed implosione di stringhe

array explode(string *separatore*, string *input* [, int *limite*])

```
$email_array = explode('@', $email);
if(strtolower($email_array[0] == 'luigi.rossi')
  $to = 'marketing@strumentifranciacorta.com';
else
  $to = info@strumentifranciacorta.com';
```

string implode(array *input*, string *separatore*)

```
$email = implode($email_array, '@');
```

string strtok(string *input*, string *separatore*): preleva da ***input*** un token alla volta

```
$token = strtok($feedback, ' ');
do{
  echo "$token <br />";
  $token = strtok(' ');
} while($token != '');
```

← Prima call → due parametri

← Successive call → un solo parametro (separatore)

Prelievo di sottostringhe

`string substr(string stringa, int inizio [, int lung])`

`inizio`: indice da cui prelevare la sottostringa (se negativo → dalla fine)

`lung`: numero di caratteri da prelevare (se non specificato → fino alla fine)

```
$feedback = 'Il vostro servizio clienti è eccellente';
```

Call	Output
`substr($feedback, 3)`	`vostro servizio clienti è eccellente`
`substr($feedback, 3, 6)`	`vostro`
`substr($feedback, -20)`	`clienti è eccellente`
`substr($feedback, -20, 7)`	`clienti`

Confronto tra stringhe

`int strcmp(string stringa1, string stringa2)`

- ***`stringa1 == stringa2`*** → risultato = 0
- ***`stringa1 > stringa2`*** → risultato > 0
- ***`stringa1 < stringa2`*** → risultato < 0

`int strnatcmp(string stringa1, string stringa2)`:
confronto sulla base di un criterio 'naturale'
(http://www.naturalordersort.org)

```
echo strcmp('15', '2');
```
⇒ **`-1`**

```
echo strnatcmp('15', '2');
```
⇒ **`1`**

Versioni non sensibili alle maiuscole/minuscole (*case insensitive*):
`strcasecmp()` **`strnatcasecomp()`**

Ricerche di sottostringhe

`string strstr(string pagliaio, string ago)`

se *`ago`* trovato nel *`pagliaio`* → sottostringa dall'inizio di *`ago`*

se *`ago`* non trovato nel *`pagliaio`* → **`false`**

se più occorrenze di *`ago`* nel *`pagliaio`* → prima occorrenza di *`ago`*

Assegnamento del destinatario in base a parole chiave nel feedback:

```
$to = 'feedback@strumentifranciacorta.com';  // default
if(strstr($feedback, 'negozio'))
  $to = 'vendite@strumentifranciacorta.com';
elseif(strstr($feedback, 'consegna'))
  $to = 'spedizioni@strumentifranciacorta.com';
elseif(strstr($feedback, 'fattura'))
  $to = 'amministrazione@strumentifranciacorta.com';
```

Varianti:

`stristr()`: case insensitive

`strrchr()` → sottostringa partendo dall'ultima occorrenza di *`ago`*

Posizione di sottostringhe

`int strpos(string pagliaio, string ago [ , int offset ])` →
restituisce la posizione della prima occorrenza di ***`ago`*** in ***`pagliaio`***
`offset`: punto in ***`pagliaio`*** da cui iniziare a cercare
se ***`ago`*** non si trova nel ***`pagliaio`*** → **`false`**
più veloce di **`strstr()`**

```
$saluto = 'Hello world';
echo strpos($saluto, 'world');
```

⇒ **6**

```
echo strpos($saluto, 'l', 5);
```

⇒ **9**

Se ***`ago`*** non si trova nel ***`pagliaio`*** → **`false`**: **`strpos()`** rimane non definito

```
$pos = strpos($saluto, 'Hello');
if($pos === false)
  echo 'Non trovato';
else
  echo "Trovato alla posizione $pos";
```

⇒ **Trovato alla posizione 0**

Sostituzione di sottostringhe (I)

mixed str_replace(mixed *ago*, mixed *nuovo*, mixed *pagliaio*
[, int &*tot*])

sostituisce tutte le occorrenze di ***ago*** in ***pagliaio***
restituisce il nuovo ***pagliaio***
se specificato, assegna a ***tot*** il numero di sostituzioni effettuate

```
$saluto = 'Hello world, wonderful world!';
$tot = 0;
echo str_replace('world', 'people', $saluto, $tot)."($tot)";
```

Possibile passare array invece di stringhe:

```
$feedback = 'Questo negozio fa schifo! Veramente schifo!';
$censurate = array('schifo', ...);
echo str_replace($censurate, '***', $feedback);
```

Sostituzione di sottostringhe (I)

mixed str_replace(mixed *ago*, mixed *nuovo*, mixed *pagliaio*
[, int &*tot*])

sostituisce tutte le occorrenze di ***ago*** in ***pagliaio***
restituisce il nuovo ***pagliaio***
se specificato, assegna a ***tot*** il numero di sostituzioni effettuate

```
$saluto = 'Hello world, wonderful world!';
$tot = 0;
echo str_replace('world', 'people', $saluto, $tot)."($tot)";
```

```
Hello people, wonderful people! (2)
```

Possibile passare array invece di stringhe:

```
$feedback = 'Questo negozio fa schifo! Veramente schifo!';
$censurate = array('schifo', ...);
echo str_replace($censurate, '***', $feedback);
```

```
Questo negozio fa ***! Veramente ***!
```

Sostituzione di sottostringhe (III)


```
string substr_replace(string stringa, string sostituto,
                      int inizio  [ , int lung ])
```

sostituisce una parte di ***stringa*** con ***sostituto***

inizio: posizione di ***stringa*** in cui inizia la sostituzione (se negativo, posizione dalla fine)

lung positiva o nulla → numero di caratteri sostituiti

lung negativa → posizione (dalla fine) su cui terminare la sostituzione

```
$saluto = 'Hello world';
echo substr_replace($saluto, 'people', 6, 5);
```

Hello people

```
echo substr_replace($saluto, 'people ', 6, 0);
```

Hello people world

```
echo substr_replace($saluto, 'people ', 6, -2);
```

Hello people ld

Espressioni regolari

Pattern matching complessi (non semplicemente uguaglianze di stringhe)

Due possibili stili: POSIX, Perl (PCRE)

- ✓ **Espressione regolare** = notazione per specificare un pattern in un testo
- ✓ Uso di **metacaratteri** per specificare particolari pattern

Insiemi di caratteri (I)

Un qualsiasi carattere diverso da \n: **.**

`.emo` ⟹ compatibile con `remo`, `temo`, `memo`, …

Classe di caratteri: **[** *lista-di-caratteri* **]**

`[aeiou]` ⟹ vocali

`[a-z]` ⟹ range delle lettere alfabetiche minuscole

`[a-zA-Z]` ⟹ range delle lettere alfabetiche

`[^a-z]` ⟹ qualsiasi carattere che non sia una lettera alfabetica minuscola

`[^aeiou]` ⟹ qualsiasi carattere che non sia una lettera vocale

Insiemi di caratteri (II)

Classi di caratteri **predefinite**:

Classe	Matches	Espansione
[[:alnum:]]	Caratteri alfanumerici	[0-9a-zA-Z]
[[:alpha:]]	Caratteri alfabetici	[a-zA-Z]
[[:ascii:]]	Caratteri ASCII (7 bit)	[\x01-\x7F]
[[:blank:]]	Spaziatura orizzontale	[\t]
[[:cntrl:]]	Caratteri di controllo	[\x01-\x1F]
[[:digit:]]	Cifre decimali	[0-9]
[[:graph:]]	Caratteri stampabili con inchiostro	[^\x01-\x20]
[[:lower:]]	Lettere minuscole	[a-z]
[[:print:]]	Caratteri stampabili (graph, spazio e \t)	[\t\x20-\xFF]
[[:punct:]]	Punteggiatura	[-!"#$%&'()*+,./:;<=>?@[\\\]^_`{|}~]
[[:space:]]	Caratteri bianchi (incluso \x0B = tab verticale)	[\n\r\t \x0B]
[[:upper:]]	Lettere maiuscole	[A-Z]
[[:xdigit:]]	Cifre esadecimali	[0-9a-fA-F]

Ripetizione e alternativa

Ripetizione zero o più volte del pattern: *****

```
[aeiou]*
```
⇨ Zero o più vocali

```
[[:alpha:]][[:alnum:]]*
```
⇨ Lettera seguita da zero o più caratteri alfanumerici (identificatore)

Ripetizione una o più volte del pattern: **+**

```
[[:digit:]]+
```
⇨ Una o più cifre decimali (numero intero)

```
[[:digit:]]+,[[:digit:]]+
```
⇨ Numero reale con parte decimale

Opzionalità: **?**

```
[[:digit:]]+[aeiou]?
```
⇨ Numero eventualmente seguito da una vocale

Alternativa: **|**

```
org|net|com
```

Sottoespressioni

Possibile raggruppare sottoespressioni mediante parentesi:

```
(molto )*grande
```

```
grande
molto grande
molto molto grande …
```

Ripetizione di un pattern un numero vincolato di volte: { *limiti* }

```
[[:digit:]]{16}
```

Numero intero di 16 cifre decimali

```
(molto ){1,3}grande
```

```
molto grande
molto molto grande
molto molto molto grande
```

```
(molto ){2,}grande
```

```
molto molto grande
molto molto molto grande
molto molto molto molto grande
```

Ancoraggio

Possibile specificare la posizione di una sottostringa:

```
^Hello
```

⇨

```
Hello world
Hello people
Hello everybody
```

```
world$
```

⇨

```
Hello world
Beutfiful world
Best world
```

Ancora	Matches
^	Inizio stringa
$	Fine stringa
[[:<:]]	Inizio parola
[[:>:]]	Fine parola

```
^[[:alpha:]][[:alnum:]]*$
```

⇨ Stringa costituita unicamente da un identificatore

```
[[:<:]][[:digit:]]+[[:>:]]
```

⇨

```
123 alfa25 460
```

Caratteri speciali

Per inibire i caratteri speciali al di fuori di [ ], è necessario precederli con \

Meglio inserire il pattern tra apici singoli (non doppi):

```
'[[:digit:]]+\.[[:digit:]]+'
```

⟹

```
12.0
0.14
13.156
```

Se pattern tra apici doppi → parsing della stringa → necessari ulteriori \

```
"[[:digit:]]+\\.[[:digit:]]+"
```

Modulo di feedback

Rilevazione di particolari termini nel feedback del cliente:

```
negozio|consegna|fattura
```

Controllo nome del cliente:

```
Luigi Rossi
G. Luigi Rossi
G.C. Luigi Rossi
```

```
^(([[:upper:]]\.)+ )?[[:upper:]][[:lower:]]+ [[:upper:]][[:lower:]]+$
```

Controllo formato indirizzo di email:

```
Luigi.Rossi-89@domain-191.unisv.it
```

```
^[a-zA-Z0-9_.-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9_.-]+$
```

Sottostringhe ed espressioni regolari

`bool ereg(string pattern, string stringa)`

- ✓ stabilisce se ***`stringa`*** contiene una sottostringa compatibile con ***`pattern`***
- ✓ case-sensitive

Variante case-insensitive: **`eregi()`**

```
if(!eregi('^[a-zA-Z0-9_.-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9_.-]+$', $email))
{
  echo '<p>Indirizzo di email non valido: torna alla pagina precedente</p>';
  exit;
}
$to = 'info@strumentifranciacorta.com';
if(eregi('negozio|servizio', $feedback))
  $to = 'vendite@strumentifranciacorta.com';
elseif(eregi('spedizione|adempimento', $feedback))
  $to = 'spedizioni@strumentifranciacorta.com';
elseif(eregi('conto|biglietto|fattura', $feedback))
  $to = 'amministrazione@strumentifranciacorta.com';
if(eregi('luigi.rossi@alice.it', $email))
  $to = 'marketing@strumentifranciacorta.com';
```

Sottostringhe ed espressioni regolari

string ereg_replace(string *pattern*, string *sostituto*, string *stringa*)

- ✓ sostituisce con ***sostituto*** tutte le sottostringhe di ***stringa*** compatibili con ***pattern***
- ✓ restituisce il risultato della sostituzione (funzionale: ***stringa*** non cambia!)

Variante case-insensitive: **eregi_replace()**

```
$transazione = 'Carta di credito: 4367-2234-1245-3200';
$pattern = '[[:digit:]]{4}(-[[:digit:]]{4}){3}';
echo ereg_replace($pattern, 'xxxx-xxxx-xxxx-xxxx', $transazione);
```

```
Carta di credito: xxxx-xxxx-xxxx-xxxx
```

```
$stringa = 'alfa 23beta gamma48 125';
$pattern = '[[:<:]][[:digit:]]+[[:>:]]';
echo ereg_replace($pattern, 'NUM', $stringa);
```

```
alfa 23beta gamma48 NUM
```

Sottostringhe ed espressioni regolari

`array split(string pattern, string stringa [, int max])`

- ✓ restituisce l'array di stringhe ottenuto dividendo ***`stringa`*** limitata dai confini compatibili con ***`pattern`***
- ✓ ***`max`***: numero massimo di elementi nell'array risultante

```
$email = 'luigi.rossi@alice.it';
$a = split('\.|@', $email);
```

⇨

$a

luigi
rossi
alice
it

```
$espressione = '3*52+24/5-12';
$operandi = split('[-+*/]', $espressione);
```

⇨

$operandi

3
52
24
5
12

Funzioni *deprecated*

`ereg('pattern','stringa')` → sostituito da
`preg_match('/pattern/','stringa')`

`eregi('pattern','stringa')` → sostituito da
`preg_match('/pattern/i','stringa')`

`ereg_replace('pattern','sostituto','stringa')` → sostituito da
`preg_replace('/pattern/','sostituto','stringa')`
uso di `'/pattern/i'` in caso di `eregi_replace`

`split('pattern','stringa')` → sostituito da
`preg_split('/pattern/','stringa')`

PROGRAMMAZIONE WEB E SERVIZI DIGITALI

PHP APPROFONDIMENTI

ARRAY E STRINGHE

Prof. Ada Bagozi
ada.bagozi@unibs.it