

PHP

Sommario

- Introduzione
- Elaborazione di stringhe e Espressioni regolari
- Variabili di ambiente Client/Server
- Elaborazione Form
- Verifica di Username e Password
- Connessione a Database
- Cookies
- Contenuti dinamici
- Precedenza operatori
- Web Resources

Obiettivi

- Gestire i tipi di dati, gli operatori, gli array e le strutture di controllo di PHP
- Capire l'elaborazione di stringhe e le espressioni regolari
- Costruire programmi per elaborare dati
- Essere in grado di leggere/scrivere dati client mediante cookie
- Costruire programmi per interagire con database MySQL

Introduzione (1)

- Il nome originale deriva da "Personal Home Page tools"
- La comunità di sviluppatori PHP ha poi modificato il nome in modo ricorsivo
 - PHP: Hypertext Preprocessor
- È Open-source
 - Chiunque può leggere, studiare, modificare e redistribuire il codice sorgente
 - È continuamente evoluto dalla comunità PHP

Introduzione (2)

- È una tecnologia per la programmazione di script sul lato server
- È indipendente dalla piattaforma

PHP

5

Generalità (1)

- Elementi di base
 - Delimitatori di script
 - Ogni script inizia con `<? php`
 - Ogni script finisce con `?>`
 - Devono racchiudere tutto il codice di script
 - Le variabili sono precedute dal simbolo `$`
 - Case-sensitive
 - Il simbolo di fine istruzione è il punto e virgola `;`

PHP

6

Generalità (2)

- Commenti
 - Se il commento è su un'unica riga il simbolo di inizio commento è `//`
 - Non c'è alcun simbolo di fine commento
 - Se il commento è su più righe
 - Inizio commento `/*`
 - Fine commento `*/`
- Per convenzione i file hanno estensione `.php`

PHP

7

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3
4 <!-- Fig. 26.1: first.php -->
5 <!-- Our first PHP script -->
6
7 <?php
8     $name = "Funatic"; // declaration
9 ?>
10
11 <html xmlns = "http://www.w3.org/1999/xhtml" >
12 <head>
13 <title>A simple PHP document</title>
14 </head>
15
16 <body style = "font-size: 2em">
17 <p>
18 <strong>
19
20 <!-- print variable name's value -->
21 Welcome to PHP, <?php print( "$name" ); ?>
22 </strong>
23 </p>
24 </body>
25 </html >
```

Scripting delimiters

Declare variable \$name

Single-line comment

Function print outputs the value of variable \$name

PHP

8

Esecuzione



Variabili (1)

- PHP è un linguaggio debolmente tipizzato
 - Una variabile può essere di tipo diverso in momenti diversi
 - Nomi di variabili all'interno di stringhe sono sostituiti dal loro valore
- Conversioni di tipo
 - settype function
 - type casting
- Operatore di concatenazione tra stringhe
 - punto .

Variabili (2)

Data type	Description
int, integer	Whole numbers (i.e., numbers without a decimal point).
float, double	Real numbers (i.e., numbers containing a decimal point).
string	Text enclosed in either single (' ') or double (" ") quotes.
bool, Boolean	True or false.
array	Group of elements of the same type.
object	Group of associated data and methods.
Resource	An external data source.
NULL	No value.

Fig. 26.2 PHP data types.

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <!-- Fig. 26.3: data.php -->
5 <!-- Demonstration of PHP data types -->
6
7 <html xmlns="http://www.w3.org/1999/xhtml">
8 <head>
9 <title>PHP data types</title>
10 </head>
11
12 <body>
13
14 <?php
15
16 // declare a string, double and Integer
17 $testString = "3.5 seconds";
18 $testDouble = 79.2;
19 $testInteger = 12;
20
21 ?>
    
```

Assign a string to variable \$testString

Assign a double to variable \$testDouble

Assign an integer to variable \$testInteger

```

22 <!-- print each variable's value -->
23 <?php print( $testString ); ?> Is a string.<br />
24 <?php print( $testDouble ); ?> Is a double.<br />
25 <?php print( $testInteger ); ?> Is an Integer.<br />
26
27 <br />
28 Now, converting to other types:<br />
29 <?php

```

Print each variable's value

```

31 // call function settype to convert variable
32 // testString to different data types
33 print( "testString" );
34 settype( $testString, "double" );
35 print( " as a double is $testString <br />" );
36 print( "testString" );
37 settype( $testString, "integer" );
38 print( " as an integer is $testString <br />" );
39 settype( $testString, "string" );
40 print( "Converting back to a string results in
41 $testString <br /><br />" );
42
43 $data = "98.6 degrees";

```

Call function settype to convert the data type of variable \$testString to a double.

Call function settype to convert the data type of variable \$testString to an integer.

Convert variable \$testString back to a string

PHP

13

```

44
45 // use type casting to cast variables to a
46 // different type
47 print( "Now using type casting instead: <br />
48 As a string - " . (string) $data .
49 <br />As a double - " . (double) $data .
50 <br />As an integer - " . (integer) $data );
51
52 ?>
53 </body>
</html >

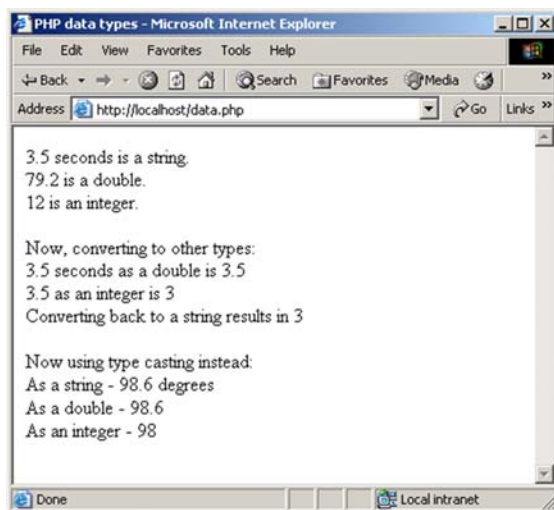
```

Use type casting to cast variable \$data to different types

PHP

14

Esecuzione



PHP

15

Operatori aritmetici

- Operatori di assegnamento
 - Prima del primo assegnamento, le variabili valgono **undef**
- Costanti
 - Sono valori a cui è associato un nome
 - Funzione define

PHP

16

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <!-- Fig. 26.4: operators.php -->
5 <!-- Demonstration of operators -->
6
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8 <head>
9 <title>Using arithmetic operators</title>
10 </head>
11
12 <body>
13 <?php
14     $a = 5;
15     print( "The value of variable a is $a <br />" );
16
17     // define constant VALUE
18     define( "VALUE", 5 );
19
20     // add constant VALUE to variable $a
21     $a = $a + VALUE;
22     print( "Variable a after adding constant VALUE
23         is $a <br />" );
24

```

Define constant VALUE.

Add constant VALUE to variable \$a.

```

25 // multiply variable $a by 2
26 $a *= 2;
27 print( "Multiplying variable a by 2 yields $a <br />" );
28
29 // test if variable $a is less than 50
30 if ( $a < 50 )
31     print( "Variable a is less than 50 <br />" );
32
33 // add 40 to variable $a
34 $a += 40;
35 print( "Variable a after adding 40 is $a <br />" );
36
37 // test if variable $a is 50 or less
38 if ( $a <= 50 )
39     print( "Variable a is still 50 or less <br />" );
40
41 // test if variable $a is between 50 and 100, inclusive
42 elseif ( $a <= 100 )
43     print( "Variable a is now between 50 and 100,
44         inclusive <br />" );
45 else
46     print( "Variable a is now greater than 100
47         <br />" );
48

```

Multiply variable \$a by two using the multiplication assignment operator *+=.

Test whether variable \$a is less than 50

Print if variable \$a is less than 50.

Add 40 to variable \$a using the addition assignment operator +=.

```

49 // print an uninitialized variable
50 print( "Using a variable before initializing:
51     $nothing <br />" );
52
53 // add constant VALUE to an uninitialized variable
54 $test = $num + VALUE;
55 print( "An uninitialized variable plus constant
56     VALUE yields $test <br />" );
57
58 // add a string to an integer
59 $str = "3 dollars";
60 $a += $str;
61 print( "Adding a string to variable a yields $a
62     <br />" );
63
64 </body>
65 </html>

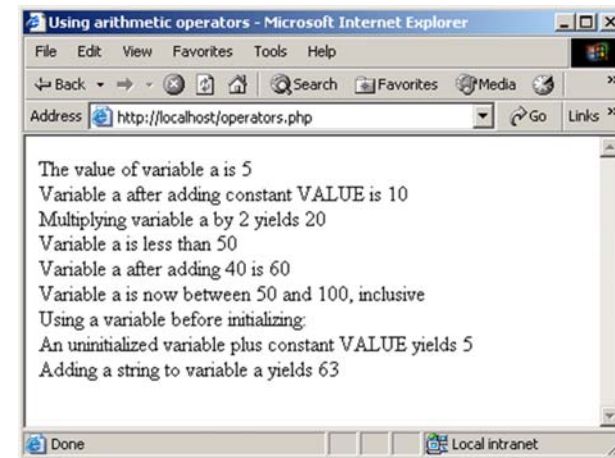
```

Print an uninitialized variable (\$nothing).

Add constant VALUE to an uninitialized variable.

Add a string to an integer.

Esecuzione



Array (1)

- Nome della variabile, seguito dall'indice racchiuso tra parentesi quadre
 - Gli indici partono da 0
- Funzioni
 - count
 - array

PHP

21

Array (2)

- Esistono costrutti predefiniti del linguaggio per la iterazione nell'array
 - reset
 - key
 - next
 - foreach loops
- Mantengono un puntatore all'elemento correntemente riferito

PHP

22

Keywords

PHP keywords					
and	do	for	include	require	true
break	else	foreach	list	return	var
case	elseif	function	new	static	virtual
class	extends	global	not	switch	xor
continue	false	if	or	this	while
default					

Fig. 26.5 PHP keywords.

PHP

23

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <!-- Fig. 26.6: arrays.php -->
5 <!-- Array manipulation -->
6
7 <html xmlns="http://www.w3.org/1999/xhtml">
8 <head>
9 <title>Array manipulation</title>
10 </head>
11
12 <body>
13 <?php
14
15 // create array first
16 print( "<strong>Creating the first array</strong>
17 <br />" );
18 $first[ 0 ] = "zero";
19 $first[ 1 ] = "one";
20 $first[ 2 ] = "two";
21 $first[] = "three";
22
23 // print each element's index and value
24 for ( $i = 0; $i < count( $first ); $i++ )
25 print( "Element $i is $first[$i] <br />" );
```

Create the array \$first by assigning a value to an array element.

Assign a value to the array, omitting the index. Appends a new element to the end of the array.

Use a for loop to print out each element's index and value. Function count returns the total number of elements in the array.

PHP

24

```

26
27 print( "<br /><strong>Creating the second array
28 </strong><br />" );
29
30 // call function array to create array second
31 $second = array( "zero", "one", "two", "three" );
32 for ( $i = 0; $i < count( $second ); $i++ )
33     print( "Element $i is $second[$i] <br />" );
34
35 print( "<br /><strong>Creating the third array
36 </strong><br />" );
37
38 // assign values to non-numerical indices
39 $third[ "ArtTic" ] = 21;
40 $third[ "LunaTic" ] = 18;
41 $third[ "GalAnt" ] = 23;
42
43 // Iterate through the array elements and print each
44 // element's name and value
45 for ( reset( $third ); $element = key( $third );
46     next( $third ) )
47     print( "$element is $third[$element] <br />" );
48

```

Call function array to create an array that contains the arguments passed to it. Store the array in variable \$second.

Assign values to non-numerical indices in array \$third.

Function reset sets the internal pointer to the first element of the array.

Function key returns the index of the element which the internal pointer references.

Function next moves the internal pointer to the next PHP element.

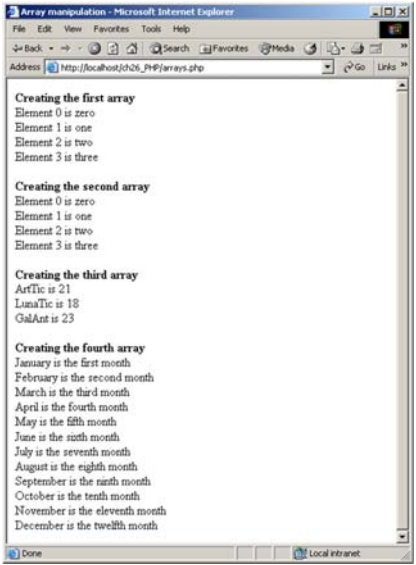
```

49 print( "<br /><strong>Creating the fourth array
50 </strong><br />" );
51
52 // call function array to create array fourth using
53 // string indices
54 $fourth = array(
55     "January" => "first", "February" => "second",
56     "March" => "third", "April" => "fourth",
57     "May" => "fifth", "June" => "sixth",
58     "July" => "seventh", "August" => "eighth",
59     "September" => "ninth", "October" => "tenth",
60     "November" => "eleventh", "December" => "twelfth"
61 );
62
63 // print each element's name and value
64 foreach ( $fourth as $element => $value )
65     print( "$element is the $value month <br />" );
66
67 ?>
68 </body>
69 </html >

```

Operator => is used in function array to assign each element a string index. The value to the left of the operator is the array index, and the value to the right is the element's value.

Esecuzione



Elaborazione di stringhe

- Funzione strcmp
 - restituisce
 - -1 se string 1 < string 2
 - 0 se string 1 = string 2
 - +1 se string 1 > string 2

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <!-- Fig. 26.7: compare.php -->
5 <!-- String Comparison -->
6
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8 <head>
9 <title>String Comparison</title>
10 </head>
11
12 <body>
13 <?php
14
15 // create array fruits
16 $fruits = array( "apple", "orange", "banana" );
17
18 // Iterate through each array element
19 for ( $i = 0; $i < count( $fruits ); $i++ ) {
20
21 // call function strcmp to compare the array element
22 // to string "banana"
23 if ( strcmp( $fruits[ $i ], "banana" ) < 0 )
24 print( $fruits[ $i ]. " is less than banana " );

```

Use a for loop to iterate through each array element.

Function strcmp compares two strings. If the first string alphabetically precedes the second, then -1 is returned. If the strings are equal, 0 is returned. If the first string alphabetically follows the second, then 1 is returned.

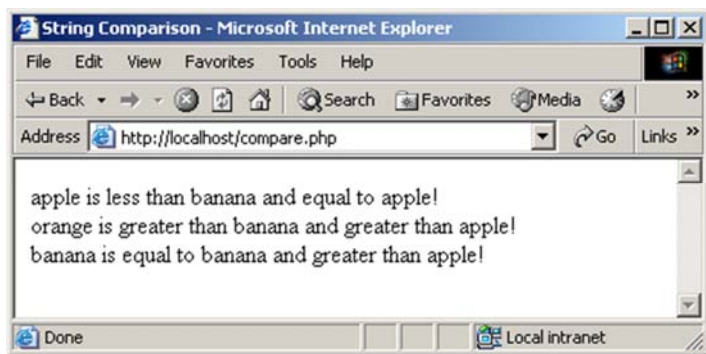
```

25 else if ( strcmp( $fruits[ $i ], "banana" ) > 0 )
26 print( $fruits[ $i ].
27 " is greater than banana " );
28 else
29 print( $fruits[ $i ]. " is equal to banana " );
30
31 // use relational operators to compare each element
32 // to string "apple"
33 if ( $fruits[ $i ] < "apple" )
34 print( "and less than apple <br />" );
35 else if ( $fruits[ $i ] > "apple" )
36 print( "and greater than apple <br />" );
37 else if ( $fruits[ $i ] == "apple" )
38 print( "and equal to apple <br />" );
39
40 }
41 ?>
42 </body>
43 </html >

```

Use relational operators to compare each array element to string "apple".

Esecuzione



Espressioni regolari

- Template per il pattern matching
 - Funzione ereg
 - POSIX
 - Funzione preg_match
 - Perl
 - Funzione ereg_replace
- Per costruire espressioni regolari
 - Metacaratteri (\$, ., ^)
 - Parentesi quadre ([,])

Metacaratteri (1)

- `.` indica qualsiasi carattere (escluso un 'a capo')
- `*` indica zero o più occorrenze (di un carattere o di un gruppo di caratteri)
- `?` indica zero o una occorrenza (di un carattere o di un gruppo di caratteri)
- `{ }` le parentesi graffe, indicano il numero esatto, o minimo, o massimo, o l'intervallo di occorrenze (di un carattere o di un gruppo di caratteri)

Metacaratteri (2)

- `+` indica una o più occorrenze (di un carattere o di un gruppo di caratteri)
- `^` indica l'inizio della stringa (o, se all'interno di una classe di caratteri, la negazione della stessa)
- `$` indica la fine della stringa
- `|` indica l'operatore OR

Metacaratteri (3)

- `\` il carattere di escape dei caratteri speciali (es. `\?` per riferirsi al punto interrogativo inteso come carattere e non come carattere speciale)
- `()` le parentesi tonde, destinate a contenere una sottostringa
- `[]` le parentesi quadre, destinate a contenere una 'classe' di caratteri

Classi di caratteri (1)

Le parentesi quadre `[]`, racchiudono una "classe di caratteri": il modello può o deve contenere alcuni o tutti i caratteri in esse contenute. Esempi:

`[abc]`

questo modello è soddisfatto quando viene trovata una delle lettere, senza tener conto dell'ordine in cui sono presenti;

`[a-z]`

in questo modello è presente un intervallo di caratteri (notare il segno `-`, sta per "dalla a alla z"), esso è soddisfatto quando viene trovato uno qualsiasi dei caratteri compresi nell'intervallo;

`[0-9]`

in questo modello è presente invece un intervallo di numeri, esso è soddisfatto quando viene trovato uno qualsiasi dei numeri compresi nell'intervallo;

`[a-z0-9\?]`

questo modello è leggermente più complesso, ma dovrebbe essere di facile comprensione. La corrispondenza viene trovata quando la stringa contiene una lettera (minuscola in questo caso), un numero o il carattere `?` (notate il segno `\` prima di `?`, perchè il punto interrogativo è un carattere speciale, che qui però assumiamo per il suo valore letterale);

`[^a-z]`

questo modello è soddisfatto quando viene trovato un qualsiasi carattere che non sia una lettera minuscola (notate il segno `^` che all'interno della classe, la nega);

Classi di caratteri (2)

- **[[:alpha:]]** indica qualsiasi lettera, maiuscola o minuscola
- **[[:digit:]]** indica qualsiasi cifra
- **[[:space:]]** indica tutti i caratteri di spazio (\t\r\n)
- **[[:upper:]]** indica le lettere maiuscole
- **[[:lower:]]** indica le lettere minuscole
- **[[:punct:]]** indica i caratteri di punteggiatura
- **[[:xdigit:]]** indica i valori esadecimali

Classi di caratteri (3)

una classe di caratteri può essere seguita (e normalmente lo è) da uno dei metacaratteri che indicano il numero di volte in cui uno dei caratteri in essa contenuti, deve essere presente, ad esempio:

[a-z0-9\?]?

i caratteri contenuti nella classe devono essere presenti zero o una volta;

[a-z0-9\?]*

i caratteri contenuti nella classe devono essere presenti zero o più volte;

[a-z0-9\?]{3}

i caratteri contenuti nella classe devono essere presenti esattamente tre volte;

[a-z0-9\?]{1,3}

i caratteri contenuti nella classe devono essere presenti da una a tre volte;

[a-z0-9\?]{3,}

i caratteri contenuti nella classe devono essere presenti minimo tre volte;

[a-z0-9\?]{,3}

i caratteri contenuti nella classe devono essere presenti massimo tre volte.

Parentesi Graffe

Indicano il numero esatto, minimo, massimo o l'intervallo di volte in cui una un'esatta sequenza o una classe di caratteri, devono essere presenti in una stringa:

- **{3}** esattamente 3 volte;
- **{3,}** minimo 3 volte;
- **{,3}** massimo 3 volte;
- **{1,3}** da 1 a 3 volte;

Parentesi Tonde

Fanno riferimento ad una sottostringa che viene assunta per il suo esatto valore letterale:

- **(abc) vs. [abc]**: (abc) indica l'esatta sequenza di caratteri, [abc] si riferisce invece ad uno dei tre caratteri.

Possono essere combinate con i metacaratteri che indicano il numero di volte in cui la sottostringa deve ripetersi:

- **(casa)?** indica la presenza opzionale della parola casa

Abbreviazioni

Usate in relazione alle classi di caratteri usate più di frequente.

- **\d** equivale a [0-9]
- **\D** equivale a [^0-9]
- **\w** equivale a [0-9A-Za-z]
- **\W** equivale a [^0-9A-Za-z]
- **\s** equivale a [\t\n\r]
- **\S** equivale a [^ \t\n\r]

ereg(arg1, arg2)

Trova la corrispondenza di un modello (arg1) all'interno di una stringa (arg2):

ereg(string espressione_regolare, string stringa [, array regs])

ereg(arg1, arg2)

Restituisce TRUE / FALSE se viene trovata o meno la corrispondenza

Il terzo argomento, opzionale restituisce l'array che contiene tanti elementi quante sono le parti del modello poste tra parentesi tonde ritrovate nella stringa più uno che sarà costituito dall'intera stringa ritrovata, e a questo array si potrà naturalmente fare riferimento per "utilizzare" quelle parti di testo ritrovate.

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <!-- Fig. 26.8: expression-php -->
5 <!-- Using regular expressions -->
6
7 <html xmlns = "http://www.w3.org/1999/xhtml "
8 <head>
9 <title>Regular expressions</title>
10 </head>
11
12 <body>
13 <?php
14 $search = "Now is the time";
15 print( "Test string is: '$search'<br /><br />" );
16
17 // call function ereg to search for pattern 'Now'
18 // in variable search
19 if ( ereg( "Now", $search ) )
20 print( "String 'Now' was found.<br />" );
21
```

Function ereg searches for the literal characters Now inside variable \$search.

```

22 // search for pattern 'Now' in the beginning of
23 // the string
24 if (ereg("Now", $search))
25     print("String 'Now' found at beginning
26         of the line.<br />");
27
28 // search for pattern 'Now' at the end of the string
29 if (ereg("Now$", $search))
30     print("String 'Now' was found at the end
31         of the line.<br />");
32
33 // search for any word ending in 'ow'
34 if (ereg("([[:<:]]([a-zA-Z]*ow)[[:>]])", $search,
35     $match))
36     print("Word found ending in 'ow': " .
37         $match[1] . "<br />");
38
39 // search for any words beginning with 't'
40 print("Words beginning with 't' found: ");
41
42 while (ereg("([[:<:]]([[:alpha:]]+)[[:>]])",
43     $search, $match)) {
44     print($match[1] . " ");
45 }

```

The special bracket expressions [[:<:]] and [[:>]] match the beginning and end of a word, respectively.

The expression inside the parentheses, [a-zA-Z]*ow, matches any word ending in ow

Placing a pattern in parentheses stores the matched string in the array that is specified in the third argument to function ereg.

The pattern used in this example, [[:<:]]([[:alpha:]]+)[[:>]], matches any word beginning with the character t followed by one or more characters. Character class [[:alpha:]] recognizes any alphabetic character.

Function eregi is used to specify case insensitive pattern matches.

The while loop is used to find each occurrence of a word in the string beginning with t.

```

46 // remove the first occurrence of a word beginning
47 // with 't' to find other instances in the string
48 $search = ereg_replace($match[1], "", $search);
49 }
50
51 print("<br />");
52 ?>
53 </body>
54 </html >

```

After printing a match of a word beginning with t, function ereg_replace is called to remove the word from the string. This is necessary because to find multiple instances of a given pattern, the first matched instance must first be removed. Function ereg_replace takes three arguments: the pattern to match, a string to replace the matched string and the string to search.

Esecuzione



Espressioni regolari: Quantificatori

Quantifier	Matches
{n}	Exactly n times.
{m, n}	Between m and n times inclusive.
{n, }	n or more times.
+	One or more times (same as {1, }).
*	Zero or more times (same as {0, }).
?	Zero or one time (same as {0, 1}).

Fig. 26.9 Some PHP quantifiers.

Espressioni regolari: Classi di caratteri

Character class	Description
al num	Alphanumeric characters (i.e., letters [a-zA-Z] or digits [0-9]).
al pha	Word characters (i.e., letters [a-zA-Z]).
di gi t	Digits.
space	Whitespace.
l ower	Lowercase letters.
upper	Uppercase letters.

Fig. 26.10 Some PHP character classes.

Variabili di ambiente Client/Server (1)

- Forniscono informazioni riguardo l'ambiente di esecuzione
 - Web browser
 - Server
 - Dettagli sulla connessione HTTP
- PHP gestisce queste informazioni in un array
 - \$_ENV

Variabili di ambiente Client/Server (2)

Variable name	Description
\$_SERVER	Data about the currently running server.
\$_ENV	Data about the client's environment.
\$_GET	Data posted to the server by the get method.
\$_POST	Data posted to the server by the post method.
\$_COOKIE	Data contained in cookies on the client's computer.
\$GLOBALS	Array containing all global variables.

Fig. 26.11 Some useful global arrays.

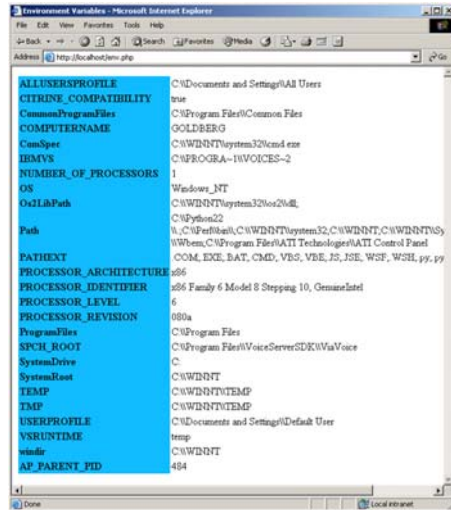
```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <!-- Fig. 26.11: env.php -->
5 <!-- Program to display environment variables -->
6
7 <html xmlns = "http://www.w3.org/1999/xhtml" >
8 <head>
9 <title>Environment Variables</title>
10 </head>
11
12 <body>
13 <table border = "0" cell padding = "2" cell spacing = "0"
14 width = "100%">
15 <?php
16
17 // print the key and value for each element
18 // in the $_ENV array
19 foreach ( $_ENV as $key => $value )
20 print( "<tr><td bgcolor = '#11bbff'>
21 <strong>$key</strong></td>
22 <td>$value</td></tr>" );
23
24 </table>
25 </body>
26 </html >
    
```

PHP stores environment variables and their values in the \$_ENV array.

The foreach loop is used to print out the keys and values for each element in the \$_ENV array.

Esecuzione



PHP

53

Elaborazione di Form

- Sono elaborati principalmente mediante
 - Proprietà action
 - Specifica dove inviare i dati del form
 - Proprietà method
 - Post
 - Ogni elemento ha un unico nome

PHP

54

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <!-- Fig. 26.13: form.html -->
5 <!-- Form for use with the form.php program -->
6
7 <html xmlns = "http://www.w3.org/1999/xhtml" >
8 <head>
9 <title>Sample form to take user input in XHTML</title>
10 </head>
11
12 <body>
13
14 <h1>This is a sample registration form.</h1>
15 Please fill in all fields and click Register.
16
17 <!-- post form data to form.php -->
18 <form method = "post" action = "form.php">
19 <img src = "images/user.gif" alt = "User" /><br />
20 <span style = "color: blue">
21 Please fill out the fields below.<br />
22 </span>
23
    
```

The action attribute of the form element indicates that when the user clicks Register, the form data will be posted to form.php.

PHP

55

```

24 <!-- create four text boxes for user input -->
25 <img src = "images/fname.gif" alt = "First Name" />
26 <input type = "text" name = "fname" /><br />
27
28 <img src = "images/lname.gif" alt = "Last Name" />
29 <input type = "text" name = "lname" /><br />
30
31 <img src = "images/email.gif" alt = "Email" />
32 <input type = "text" name = "email" /><br />
33
34 <img src = "images/phone.gif" alt = "Phone" />
35 <input type = "text" name = "phone" /><br />
36
37 <span style = "font-size: 10pt">
38 Must be in the form (555)555-5555</span>
39 <br /><br />
40
41 <img src = "images/downloads.gif"
42 alt = "Publications" /><br />
43
44 <span style = "color: blue">
45 Which book would you like information about?
46 </span><br />
47
    
```

A unique name (e.g., email) is assigned to each of the form's input fields. When Register is clicked, each field's name and value are sent to the Web server.

PHP

56

```

48 <!-- create drop-down list containing book names -->
49 <select name = "book">
50   <option>Internet and WWW How to Program 3e</option>
51   <option>C++ How to Program 4e</option>
52   <option>Java How to Program 5e</option>
53   <option>XML How to Program 1e</option>
54 </select>
55 <br /><br />
56
57 <img src = "images/os.gif" alt = "operating System" />
58 <br /><span style = "color: blue">
59   Which operating system are you currently using?
60 <br /></span>
61
62 <!-- create five radio buttons -->
63 <input type = "radio" name = "os" value = "Windows XP"
64   checked = "checked" />
65   Windows XP
66
67 <input type = "radio" name = "os" value =
68   "Windows 2000" />
69   Windows 2000
70
71 <input type = "radio" name = "os" value =
72   "Windows 98" />
73   Windows 98<br />

```

PHP

57

```

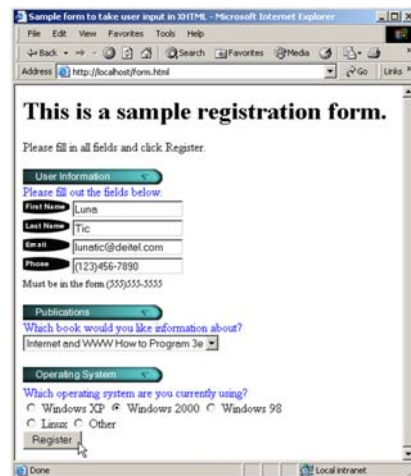
74
75   <input type = "radio" name = "os" value = "Linux" />
76     Linux
77
78   <input type = "radio" name = "os" value = "Other" />
79     Other<br />
80
81   <!-- create a submit button -->
82   <input type = "submit" value = "Register" />
83 </form>
84
85 </body>
86 </html >

```

PHP

58

Esecuzione



PHP

59

Elaborazione Server dei dati sottomessi in un form (1)

- Conferma della validità dei dati sottomessi
 - Funzione extract
 - Crea variabili corrispondenti a ogni coppia chiave-valore nell'array
 - Permette di recuperare facilmente tutti i valori inviati a una pagina PHP
- Uso di espressioni regolari

PHP

60

Elaborazione Server dei dati sottomessi in un form (2)

- Buona norma di programmazione
 - Effettuare sul lato client tutte le verifiche possibili, così da poter alleggerire le attività del server
 - JavaScript
- Fine di uno script
 - Funzione die

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <!-- Fig. 26.14: form.php -->
5 <!-- Read information sent from form.html -->
6
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8 <head>
9 <title>Form Validation</title>
10 </head>
11
12 <body style = "font-family:arial,sans-serif">
13
14 <?php
15     extract( $_POST );
16
17     // determine whether phone number is valid and print
18     // an error message if not
19     if ( !ereg( "\([0-9]{3}\)[0-9]{3}-[0-9]{4}*",
20         $phone ) ){
21

```

Function ereg is called to determine whether the phone number entered by the user is valid.

The expression \ (matches the opening parentheses of a phone number.

The parentheses in the expression must be followed by three digits ([0-9]{3}), a closing parenthesis, three digits, a literal hyphen and four additional digits.

We access the phone field's value from form.html by using variable \$phone.

```

22 print( "<p><span style = \"color: red;
23     font-size: 2em\">
24     INVALID PHONE NUMBER</span><br />
25     A valid phone number must be in the form
26     <strong>(555)555-5555</strong><br />
27     <span style = \"color: blue\">
28     Click the Back button, enter a valid phone
29     number and resubmit.<br /><br />
30     Thank You.</span></p></body></html >" );
31
32 die(); // terminate script execution
33 }
34 ?>
35
36 <p>Hi
37 <span style = "color: blue">
38 <strong>
39 <?php print( "$fname" ); ?>
40 </strong>
41 </span>.
42 Thank you for completing the survey.<br />
43

```

Function die terminates script execution

```

44 You have been added to the
45 <span style = "color: blue">
46 <strong>
47 <?php print( "$book " ); ?>
48 </strong>
49 </span>
50 mailing list.
51 </p>
52 <strong>The following information has been saved
53 in our database:</strong><br />
54
55 <table border = "0" cellpadding = "0" cellspacing = "10">
56 <tr>
57 <td bgcolor = "#ffffaa">Name </td>
58 <td bgcolor = "#ffffbb">Email </td>
59 <td bgcolor = "#ffffcc">Phone</td>
60 <td bgcolor = "#ffffdd">OS</td>
61 </tr>
62
63 <tr>
64 <?php
65

```

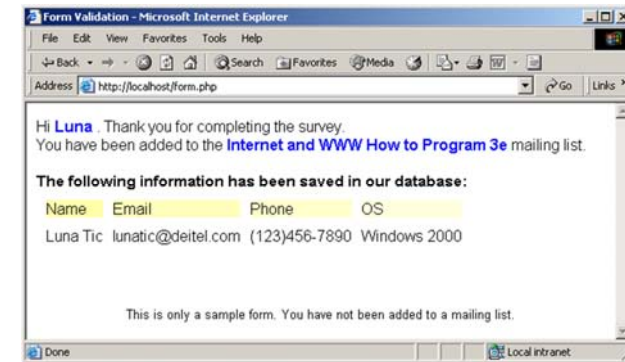


```

66 // print each form field's value
67 print( "<td>{$fname} {$lname}</td>
68 <td>{$email}</td>
69 <td>{$phone}</td>
70 <td>{$os}</td>" );
71
72 </tr>
73 </table>
74
75 <br /><br /><br />
76 <div style = "font-size: 10pt; text-align: center">
77     This is only a sample form.
78     You have not been added to a mailing list.
79 </div>
80 </body>
81 </html>

```

Esecuzione



PHP

65

PHP

66

Verifica di Username e Password (1)

- Per siti web ad accesso controllato
 - L'accesso è permesso solo a chi ne ha diritto
 - Per motivi di sicurezza i dati di username e password sono criptati quando
 - spediti,
 - memorizzati,
 - recuperati

PHP

67

Verifica di Username e Password (2)

- I dati di login sono memorizzati in un file
 - Funzione fopen, in modalità di
 - read
 - write
 - append
 - Memorizzazione mediante funzione fputs
 - \n carattere di newline
 - La chiusura del file avviene mediante la funzione fclose

PHP

68

Verifica di Username e Password (3)

- Altre funzioni utili
 - Funzione chop
 - Elimina il carattere di newline
 - Funzione split
 - Spezza la stringa in sottostringhe

PHP

69

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <!-- Fig. 26.15: password.html -->
5 <!-- XHTML form sent to password.php for verification -->
6
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8 <head>
9 <title>Verifying a username and a password.</title>
10
11 <style type = "text/css">
12 td { background-color: #DDDDDD }
13 </style>
14 </head>
15
16 <body style = "font-family: arial">
17 <p style = "font-size: 13pt">
18 Type in your username and password below.
19 <br />
20 <span style = "color: #0000FF; font-size: 10pt;
21 font-weight: bold">
22 Note that password will be sent as plain text
23 </span>
24 </p>
25
```

PHP

70

```
26 <!-- post form data to password.php -->
27 <form action = "password.php" method = "post">
28 <br />
29
30 <table border = "0" cell spacing = "0"
31 style = "height: 90px; width: 123px;
32 font-size: 10pt" cell padding = "0">
33
34 <tr>
35 <td colspan = "3">
36 <strong>Username:</strong>
37 </td>
38 </tr>
39
40 <tr>
41 <td colspan = "3">
42 <input size = "40" name = "USERNAME"
43 style = "height: 22px; width: 115px" />
44 </td>
45 </tr>
46
```

Form data is posted to password.php.

PHP

71

```
47 <tr>
48 <td colspan = "3">
49 <strong>Password:</strong>
50 </td>
51 </tr>
52
53 <tr>
54 <td colspan = "3">
55 <input size = "40" name = "PASSWORD"
56 style = "height: 22px; width: 115px"
57 type = "password" />
58 <br/></td>
59 </tr>
60
61 <tr>
62 <td colspan = "1">
63 <input type = "submit" name = "Enter"
64 value = "Enter" style = "height: 23px;
65 width: 47px" />
66 </td>
67 <td colspan = "2">
68 <input type = "submit" name = "NewUser"
69 value = "New User"
70 style = "height: 23px" />
71 </td>
```

PHP

72

```

72     </tr>
73   </table>
74 </form>
75 </body>
76 </html >

```

Esecuzione



```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <!-- Fig. 26.16: password.php -->
5 <!-- Searching a database for usernames and passwords. -->
6
7 <html xmlns = "http://www.w3.org/1999/xhtml" >
8 <head>
9 <?php
10 extract( $_POST );
11
12 // check if user has left USERNAME or PASSWORD field blank
13 if ( !$USERNAME || !$PASSWORD ) {
14     f i e l d s B l a n k ( ) ;
15     d i e ( ) ;
16 }
17
18 // check if the New User button was clicked
19 if ( !isset( $NewUser ) ) {
20
21     // open password.txt for writing using append mode
22     if ( ! ( $file = fopen( "password.txt",
23         "a" ) ) ) {
24

```

Variable names, when preceded by the logical negation operator (!), return true if they are empty or set to 0. This checks if a user has submitted a form without specifying a username or password.

Function f i e l d s B l a n k is called if the user has submitted an incomplete form to notify the user that all form fields must be completed.

Function i s s e t tests whether the user has pressed the **New User** button, indicating that a new user must be added.

To add a new user, we open the file password.txt in append mode and assign the file handle that is returned to variable \$file.

```

25 // print error message and terminate script
26 // execution if file cannot be opened
27 print( "<title>Error</title></head><body>
28     Could not open password file
29 </body></html >" );
30 die();
31 }
32
33 // write username and password to file and
34 // call function userAdded
35 fputs( $file, "$USERNAME,$PASSWORD\n" );
36 userAdded( $USERNAME );
37 }
38 else {
39
40     // If a new user is not being added, open file
41     // for reading
42     if ( ! ( $file = fopen( "password.txt",
43         "r" ) ) ) {
44         print( "<title>Error</title></head>
45             <body>Could not open password file
46             </body></html >" );
47         die();
48     }
49

```

Print an error message and terminate script execution if the file cannot be opened.

Function fputs writes the name and password to the text file..

Function userAdded is called to print a message to the user to indicate that the username and password were added to the file.

```

50 $userVerified = 0;
51
52 // read each line in file and check username
53 // and password
54 while ( !feof( $file ) && ! $userVerified ) {
55
56     // read line from file
57     $line = fgets( $file, 255 );
58
59     // remove newline character from end of line
60     $line = chop( $line );
61
62     // split username and password
63     $field = split( ",", $line, 2 );
64
65     // verify username
66     if ( $USERNAME == $field[ 0 ] ) {
67         $userVerified = 1;
68
69         // call function checkPassword to verify
70         // user's password
71         if ( checkPassword( $PASSWORD, $field )
72             == true )
73             accessGranted( $USERNAME );
74         else
75             wrongPassword();

```

Before entering the while loop, variable \$userVerified is set to 0.

fgets reads a line from the text file. Result is assigned to variable \$line.

The while executes as long as there are more lines in file and variable \$userVerified is still 0 or empty.

chop removes newline from the end of line.

Function split is called to separate the string at the specified delimiter (in this case, a comma). The resulting array is stored in array \$field.

The username entered by the user is tested against the one returned in the text file (stored in the first element of the array). If they match, variable \$userVerified is set to 1.

Function checkPassword is called to verify the user's password. Variable \$PASSWORD and array \$field are passed to the function.

PHP function checkPassword returns true, function accessGranted is called to notify the client that permission has been granted. Otherwise, function wrongPassword is called.

```

76     }
77 }
78
79 // close text file
80 fclose( $file );
81
82 // call function accessDenied if username has
83 // not been verified
84 if ( ! $userVerified )
85     accessDenied();
86
87
88 // verify user password and return a boolean
89 function checkPassword( $userpassword, $filedata )
90 {
91     if ( $userpassword == $filedata[ 1 ] )
92         return true;
93     else
94         return false;
95 }
96

```

After the while loop has executed, function fclose is called to close the file.

If variable \$userVerified has not been set to a value other than 0, function accessDenied is called to notify the client that access has been denied.

Function checkPassword compares the user's password to the password in the file. If they match, true is returned, whereas false is returned if they do not.

```

97 // print a message indicating the user has been added
98 function userAdded( $name )
99 {
100     print( "<title>Thank You</title></head>
101     <body style = \"font-family: arial;
102     font-size: 1em; color: blue\">
103     <strong>You have been added
104     to the user list, $name.
105     <br />Enjoy the site.</strong>\" );
106 }
107
108 // print a message indicating permission
109 // has been granted
110 function accessGranted( $name )
111 {
112     print( "<title>Thank You</title></head>
113     <body style = \"font-family: arial;
114     font-size: 1em; color: blue\">
115     <strong>Permission has been
116     granted, $name. <br />
117     Enjoy the site.</strong>\" );
118 }
119

```

Function userAdded prints a message to the client indicating that the user has been added.

Function accessGranted prints a message to the client indicating that permission has been granted.

```

120 // print a message indicating password is invalid
121 function wrongPassword()
122 {
123     print( "<title>Access Denied</title></head>
124     <body style = \"font-family: arial;
125     font-size: 1em; color: red\">
126     <strong>You entered an invalid
127     password.<br />Access has
128     been denied.</strong>\" );
129 }
130
131 // print a message indicating access has been denied
132 function accessDenied()
133 {
134     print( "<title>Access Denied</title></head>
135     <body style = \"font-family: arial;
136     font-size: 1em; color: red\">
137     <strong>
138     You were denied access to this server.
139     <br /></strong>\" );
140 }
141

```

Function wrongPassword prints a message to the client indicating that the password is invalid.

Function accessDenied prints a message to the client indicating that access has been denied.

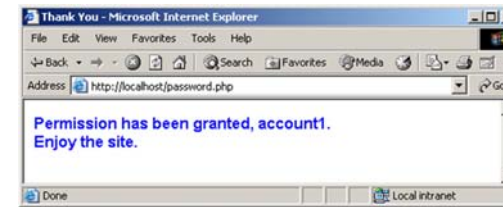
```

142 // print a message indicating that fields
143 // have been left blank
144 function fieldsBlank() ←
145 {
146     print( "<title>Access Denied</title></head>
147           <body style = \"font-family: arial;
148           font-size: 1em; color: red\">
149           <strong>
150           Please fill in all form fields.
151           <br /></strong>" );
152 }
153
154 ?>
155 </body>
156 </html >

```

Function fieldsBlank prints a message to the client indicating that all form fields have not been completed.

Esecuzione



PHP

81

PHP

82

```

1 account1, password1
2 account2, password2
3 account3, password3
4 account4, password4
5 account5, password5
6 account6, password6
7 account7, password7
8 account8, password8
9 account9, password9
10 account10, password10

```

Database

- Per database intendiamo qualunque sistema atto a memorizzare dati organizzati
- Ci concentriamo su MySQL
 - Free
 - Si interfaccia bene con PHP
 - Il linguaggio fornisce modalità per accedere al db e ai suoi dati direttamente dalle pagine Web

PHP

83

PHP

84

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <!-- Fig. 26.18: data.html -->
5 <!-- Querying a MySQL Database -->
6
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8 <head>
9 <title>Sample Database Query</title>
10 </head>
11
12 <body style = "background-color: #F0E68C">
13 <h2 style = "font-family: arial color: blue">
14 Querying a MySQL database.
15 </h2>
16
17 <form method = "post" action = "database.php">
18 <p>Select a field to display:
19
20 <!-- add a select box containing options -->
21 <!-- for SELECT query -->

```

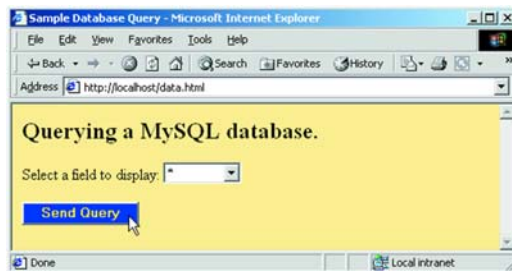
```

22 <select name = "select">
23 <option selected = "selected">*</option>
24 <option>ID</option>
25 <option>Title</option>
26 <option>Category</option>
27 <option>ISBN</option>
28 </select>
29
30 </p>
31 <input type = "submit" value = "Send Query"
32 style = "background-color: blue;
33 color: yellow; font-weight: bold" />
34 </form>
35 </body>
36 </html >

```

Select box containing options for a SELECT query.

Esecuzione



Connessione a Database

- SQL (Structured Query Language): linguaggio usato per interagire con un db
- Offre molte funzioni utili:
 - mysql_connect
 - mysql_select_db
 - mysql_query
 - mysql_error
 - mysql_fetch_row
 - mysql_close
 - ...

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <!-- Fig. 26.19: database.php -->
5 <!-- Program to query a database and -->
6 <!-- send results to the client. -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9 <head>
10 <title>Search Results</title>
11 </head>
12
13 <body style = "font-family: arial, sans-serif"
14 style = "background-color: #F0E68C">
15 <?php
16
17     extract( $_POST );
18
19     // build SELECT query
20     $query = "SELECT " . $select . " FROM Books";
21
22     // Connect to MySQL
23     if ( ! ( $database = mysql_connect( "localhost",
24 "httpd", "" ) ) )
25         die( "Could not connect to database" );

```

Build the select query and assign the string to variable \$query.

Function mysql_connect returns a database handle which represents PHP's connection to a database. If this connection is not made, function die is called to terminate script execution.

PHP

89

```

26
27 // open Products database
28 if ( !mysql_select_db( "Products", $database ) )
29     die( "Could not open Products database" );
30
31 // query Products database
32 if ( ! ( $result = mysql_query( $query, $database ) ) ) {
33     print( "Could not execute query! <br />" );
34     die( mysql_error() );
35 }
36 ?>
37
38 <h3 style = "color: blue">
39 Search Results</h3>
40
41 <table border = "1" cellpadding = "3" cellspacing = "2"
42 style = "background-color: #ADD8E6">
43
44 <?php
45
46 // fetch each record in result set
47 for ( $counter = 0;
48 $row = mysql_fetch_row( $result );
49 $counter++ ) {
50

```

Function mysql_select_db is called to specify the database to be queried.

Function mysql_query returns an object containing the result set of the query, which we assign to variable \$result.

The for loop iterates through each record in the result set while constructing an XHTML table from the results. Variable \$counter is incremented by one for each row retrieved.

Function mysql_fetch_row returns an array containing the elements of each row in the result set of our query (\$result).

PHP

90

```

51 // build table to display results
52 print( "<tr>" );
53
54 foreach ( $row as $key => $value )
55     print( "<td>$value</td>" );
56
57 print( "</tr>" );
58 }
59
60 mysql_close( $database );
61 ?>
62
63 </table>
64
65 <br />Your search yielded <strong>
66 <?php print( "$counter" ) ?> results.<br /><br /></strong>
67
68 <h5>Please email comments to
69 <a href = "mailto:delitel@delitel.com">
70 Delitel and Associates, Inc.
71 </a>
72 </h5>
73
74 </body>
75 </html>

```

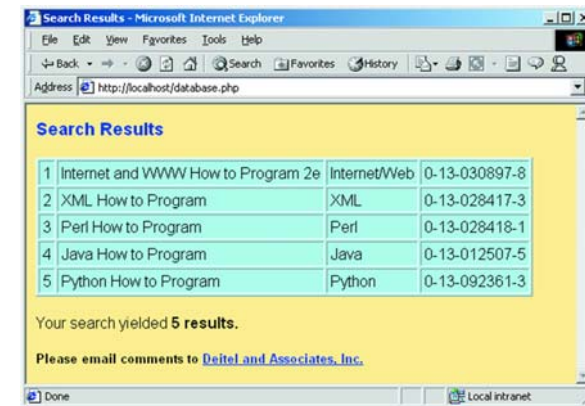
The foreach loop iterates through the array containing the elements of each row and prints out each element in an individual table cell.

The total number of results are printed to the client.

PHP

91

Esecuzione



PHP

92

Cookies (1)

- Cookies: file di testo che registrano sul client informazioni relative al client stesso
 - Evitano di ripetere informazioni precedentemente fornite, ad esempio preferenze o particolari impostazioni
- Possono rappresentare attentati alla privacy
 - Attenzione alla registrazione di dati sensibili

PHP

93

Cookies (2)

- PHP fornisce strumenti per la gestione dei cookie
 - Funzione setcookie
 - Name
 - Value
 - Expiration date

PHP

94

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <!-- Fig. 26.20: cookies.html -->
5 <!-- Writing a Cookie -->
6
7 <html xmlns = "http://www.w3.org/1999/xhtml" >
8   <head>
9     <title>Writing a cookie to the client computer</title>
10  </head>
11
12  <body style = "font-family: arial, sans-serif;
13    background-color: #99CCFF">
14
15    <h2>Click Write Cookie to save your cookie data.</h2>
16
```

PHP

95

```
17 <form method = "post" action = "cookies.php"
18   style = "font-size: 10pt">
19   <strong>Name:</strong><br />
20   <input type = "text" name = "NAME" /><br />
21
22   <strong>Height:</strong><br />
23   <input type = "text" name = "HEIGHT" /><br />
24
25   <strong>Favorite Color:</strong><br />
26   <input type = "text" name = "COLOR" /><br />
27
28   <input type = "submit" value = "Write Cookie"
29     style = "background-color: #F0E86C; color: navy;
30     font-weight: bold" /></p>
31 </form>
32 </body>
33 </html >
```

Form data is posted to cookies.php.

PHP

96

Esecuzione



PHP

97

```

1 <?php
2 // Fig. 26.21: cookies.php
3 // Program to write a cookie to a client's machine
4
5 extract( $_POST );
6 // write each form field's value to a cookie and set the
7 // cookie's expiration date
8 setcookie( "Name", $NAME, time() + 60 * 60 * 24 * 5 );
9 setcookie( "Height", $HEIGHT, time() + 60 * 60 * 24 * 5 );
10 setcookie( "Color", $COLOR, time() + 60 * 60 * 24 * 5 );
11 ?>
12
13 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
14 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
15
16 <html xmlns = "http://www.w3.org/1999/xhtml ">
17 <head>
18 <title>Cookie Saved</title>
19 </head>
20
21 <body style = "font-family: arial, sans-serif">
22 <p>The cookie has been set with the following data:</p>
23

```

Function setcookie takes the name of the cookie to be set as the first argument, followed by the value to be stored in the cookie. The optional third argument specifies the expiration date of the cookie.

PHP

98

```

24 <!-- print each form field's value -->
25 <br /><span style = "color: blue">Name:</span>
26 <?php print( $NAME ) ?><br />
27
28 <span style = "color: blue">Height:</span>
29 <?php print( $HEIGHT ) ?><br />
30
31 <span style = "color: blue">Favorite Color:</span>
32
33 <span style = "color: <?php print( "$COLOR">$COLOR" ) ?>
34 </span><br />
35 <p>Click <a href = "readCookies.php">here</a>
36 to read the saved cookie.</p>
37 </body>
38 </html >

```

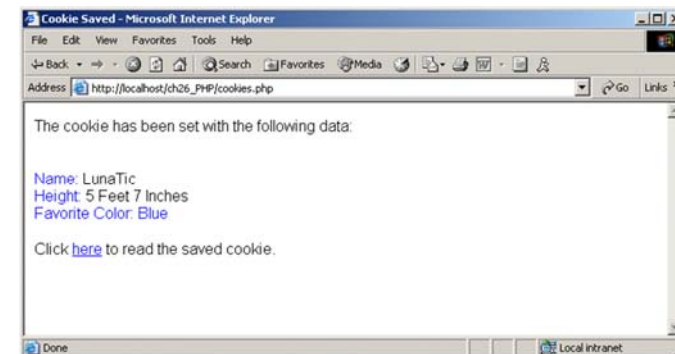
Each form field's value is printed to confirm the data that has been set as a cookie with the user.

Hyperlink to readCookies.php.

PHP

99

Esecuzione



PHP

100

Lettura di Cookie

- Variabile di ambiente `$_COOKIE`
 - Array
- È possibile accedere ad ogni elemento dell'array con il loop `foreach`
 - Divide l'elemento in due:
 - chiave
 - valore

PHP

101

Memorizzazione di Cookie (1)

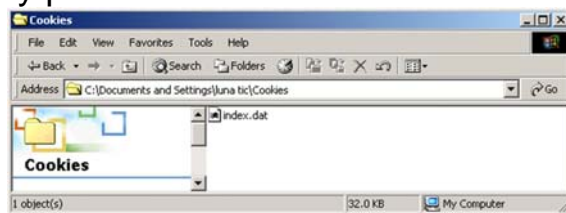
- I cookie sono memorizzati in file di testo localizzati nel file system del client in un'area nota al browser
 - Ad esempio directory Cookies per Internet Explorer

PHP

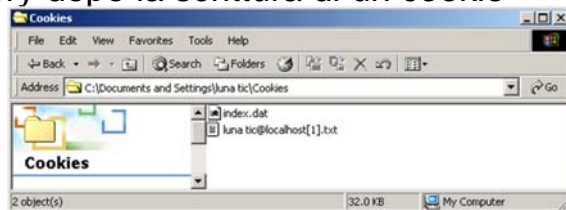
102

Memorizzazione di Cookie (2)

Directory prima della scrittura di un cookie



Directory dopo la scrittura di un cookie



PHP

103

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <!-- Fig. 26.24: readCookies.php -->
5 <!-- Program to read cookies from the client's computer -->
6
7 <html xmlns = "http://www.w3.org/1999/xhtml" >
8 <head><title>Read Cookies</title></head>
9
10 <body style = "font-family: arial, sans-serif">
11
12 <p>
13 <strong>
14 The following data is saved in a cookie on your
15 computer.
16 </strong>
17 </p>
18
```

PHP

104

```

19 <table border = "5" cel l spac ing = "0" cel l padd ing = "10">
20 <?php
21
22 // Iterate through array $_COOKIE and print
23 // name and value of each cookie
24 foreach ( $_COOKIE as $key => $value )
25     print( "<tr>
26         <td bgcolor=#F0E68C>#$key</td>
27         <td bgcolor=#FFA500>#$value</td>
28     </tr>" );
29
30
31 </table>
32 </body>
33 </html >

```

PHP creates array \$_COOKIE which contains all cookie values indexed by their names.

The foreach loop iterates through the \$_COOKIE array and prints the name and value of each cookie in an XHTML table.

Esecuzione



Contenuti Dinamici (1)

- Permettono di modificare dinamicamente il contenuto delle pagine XHTML
 - La proprietà action di un form si riferisce alla pagina che lo contiene
 - Svolge azioni diverse quando la pagina è caricata e quando il forma è inviato
 - Variabile isset

Contenuti Dinamici (2)

- Sintassi \$\$variable syntax
 - Permette di riferirsi a variabili il cui nome è il valore della variabile \$variable
- Se l'input è valido, allora effettua chiamate a un db MySQL

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <!-- Fig. 26.25: dynamicForm.php -->
5 <!-- Form for use with the form.php program -->
6
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8 <head>
9 <title>Sample form to take user input in XHTML</title>
10 </head>
11
12 <body>
13 <?php
14     extract ( $_POST );
15     $!error = false;
16
17     // array of book titles
18     $booklist = array( "Internet and WWW How to Program 3e",
19                      "C++ How to Program 4e",
20                      "Java How to Program 5e",
21                      "XML How to Program 1e" );
22

```

Build array of options for the form.

```

23 // array of possible operating systems
24 $systemlist = array( "Windows XP",
25                    "Windows 2000",
26                    "Windows 98",
27                    "Linux",
28                    "Other");
29
30 // array of name and alt values for the text input fields
31 $inputlist = array( "fname" => "First Name",
32                   "lname" => "Last Name",
33                   "email" => "Email",
34                   "phone" => "Phone" );
35
36 if ( !isset ( $submit ) ) {
37     if ( $fname == "" ) {
38         $formerrors[ "fnameerror" ] = true;
39         $!error = true;
40     }
41
42     if ( $lname == "" ) {
43         $formerrors[ "lnameerror" ] = true;
44         $!error = true;
45     }
46

```

If the page is being loaded as a result of a form submission, do error checking and then retrieve information from the database.

Check for errors or omissions in form field input.

```

47 if ( $email == "" ) {
48     $formerrors[ "emailerror" ] = true;
49     $!error = true;
50 }
51
52 if ( !ereg( "^[0-9]{3}[0-9]{3}-[0-9]{4}$", $phone ) ) {
53     $formerrors[ "phoneerror" ] = true;
54     $!error = true;
55 }
56
57 if ( !$!error ) {
58
59     // build INSERT query
60     $query = "INSERT INTO contacts " .
61             "( LastName, FirstName, Email, Phone, Book, OS ) " .
62             "VALUES ( '$lname', '$fname', '$email', " .
63             "'\" . quotemeta( $phone ) . "\" , '$book', '$os' )";
64
65     // Connect to MySQL
66     if ( ! ( $database = mysql_connect( "localhost",
67                                     "httpd", "" ) ) )
68         die( "Could not connect to database" );
69
70     // open MailingList database
71     if ( !mysql_select_db( "MailingList", $database ) )
72         die( "Could not open MailingList database" );

```

If there were no errors, query the MySQL database.

```

73
74 // execute query in MailingList database
75 if ( ! ( $result = mysql_query( $query, $database ) ) ) {
76     print( "Could not execute query! <br />" );
77     die( mysql_error() );
78 }
79
80 print( "<p>Hi
81 <span style = 'color: blue'>
82 <strong>$fname</strong></span>.
83 Thank you for completing the survey.<br />
84
85 You have been added to the
86 <span style = 'color: blue'>
87 <strong>$book</strong></span>
88 mailing list.
89 </p>
90 <strong>The following information has been saved
91 in our database:</strong><br />
92
93 <table border = '0' cellpadding = '0' cellspacing = '10' >
94 <tr>
95 <td bgcolor = '#ffffff'>Name</td>
96 <td bgcolor = '#ffffbb'>Email</td>
97 <td bgcolor = '#ffffcc'>Phone</td>

```

```

98     <td bgcolor = '#ffffdd'>0S</td>
99     </tr>
100    </tr>
101
102    <!-- print each form field's value -->
103    <td>$fname $!name</td>
104    <td>$email</td>
105    <td>$phone</td>
106    <td>$os</td>
107    </tr></table>
108
109    <br /><br /><br />
110    <div style = 'font-size: 10pt; text-align: center'>
111    <div style = 'font-size : 18pt'>
112    <a href = 'formDatabase.php'>
113    Click here to view entire database.</a></div>
114    This is only a sample form.
115    You have not been added to a mailing list.
116    </div></body></html>" );
117    die();
118    }
119    }
120
121    print( "<h1>This is a sample registration form.</h1>
122    Please fill in all fields and click Register." );

```

PHP

113

Halt the script so the form-generation code does not execute.

```

123
124    if ( $!error ) {
125        print( "<br /><span style = 'color : red'>
126            Fields with * need to be filled in properly.</span>" );
127    }
128
129    print( "<!-- post form data to form.php -->
130    <form method = 'post' action = 'dynamicform.php'>
131    <img src = 'images/user.gif' alt = 'User' /><br />
132    <span style = 'color: blue'>
133    Please fill out the fields below.<br />
134    </span>
135
136    <!-- create four text boxes for user input -->" );
137    foreach ( $!inputlist as $!inputname => $!inputval ) {
138        $!inputtext = $!inputval . $!inputname ;
139
140        print( "<img src = 'images/$!inputname.gif'
141            alt = '$!inputval' /><input type = 'text'
142            name = '$!inputname' value = '$!inputval' />" );
143
144        if ( $formerrors[ ( $!inputname ). "error" ] == true )
145            print( "<span style = 'color : red'>*</span>" );
146
147        print( "<br />" );
148    }

```

PHP

114

Fill in the forms using `$$variable` syntax.

If the form input contained errors, place a red asterisk (*) next to the text field.

```

149
150    print( "<span style = 'font-size : 10pt' );
151
152    if ( $formerrors[ "phoneerror" ] )
153        print( "<span style = 'color : red' );
154
155    print( "<span style = 'color : blue'>
156        Must be in the form (555)555-5555
157        </span><br /><br />
158
159    <img src = 'images/downloads.gif'
160    alt = 'Publications' /><br />
161
162    <span style = 'color: blue'>
163    Which book would you like information about?
164    </span><br />
165
166    <!-- create drop-down list containing book names -->
167    <select name = 'book'>
168
169    foreach ( $booklist as $currbook ) {
170        print( "<option" );
171
172        if ( ( $currbook == $book ) )
173            print( " selected = 'true'" );

```

PHP

115

Make sure the correct book is selected in the dropdown box.

```

174    print( ">$currbook</option>" );
175    }
176
177    print( "</select><br /><br />
178    <img src = 'images/os.gif' alt = 'Operating System' />
179    <br /><span style = 'color: blue'>
180    Which operating system are you currently using?
181    <br /></span>
182
183    <!-- create five radio buttons -->" );
184
185    $counter = 0;
186
187    foreach ( $systemlist as $currsystem ) {
188        print( "<input type = 'radio' name = 'os'
189            value = '$currsystem'" );
190
191        if ( $currsystem == $os ) print( "checked = 'checked'" );
192        if ( $!error && $counter == 0 ) print( "checked = 'checked'" );
193
194        print( " />$currsystem" );
195
196        if ( $counter == 2 ) print( "<br />" );
197        $counter++;
198    }
199

```

PHP

116

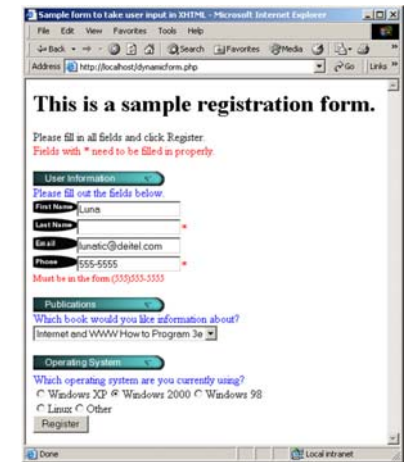
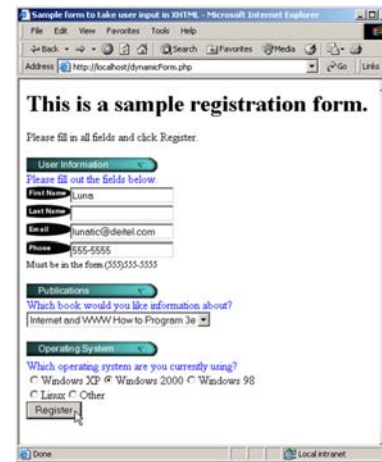
Make sure the correct OS is checked in the checkbox.

```

200 print( "<!-- create a submit button -->
201 <br />
202 <input type = 'submit' name = 'submit' value = 'Register' />
203 </form></body></html >" );
204 ?>

```

Esecuzione (1)



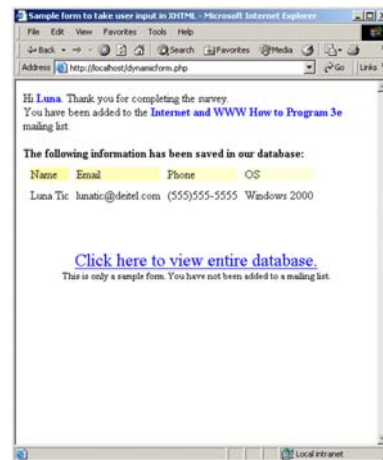
PHP

117

PHP

118

Esecuzione (2)



PHP

119

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <!-- Fig. 26.26: formDatabase.php -->
5 <!-- Program to query a database and -->
6 <!-- send results to the client. -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml ">
9 <head>
10 <title>Search Results</title>
11 </head>
12
13 <body style = "font-family: arial, sans-serif"
14 style = "background-color: #F0E68C">
15 <?php
16
17     extract( $_POST );
18
19     // build SELECT query
20     $query = "SELECT * FROM contacts";
21
22     // Connect to MySQL
23     if ( ! ( $database = mysql_connect( "localhost",
24 "httpd", "" ) ) )
25         die( "Could not connect to database" );

```

PHP

120

```

26
27 // open MailingList database
28 if ( !mysql_select_db( "MailingList", $database ) )
29     die( "Could not open MailingList database" );
30
31 // query MailingList database
32 if ( ! ( $result = mysql_query( $query, $database ) ) ) {
33     print( "Could not execute query! <br />" );
34     die( mysql_error() );
35 }
36
37
38 <h3 style = "color: blue">
39 Mailing List Contacts</h3>
40
41 <table border = "1" cellpadding = "3" cellspacing = "2"
42     style = "background-color: #ADD8E6">
43
44     <tr>
45         <td>ID</td>
46         <td>Last Name</td>
47         <td>First Name</td>
48         <td>E-mail Address</td>
49         <td>Phone Number</td>
50         <td>Book</td>

```

PHP

121

```

51     <td>Operating System</td>
52 </tr>
53 <?php
54
55     // fetch each record in result set
56     for ( $counter = 0;
57         $row = mysql_fetch_row( $result );
58         $counter++){
59
60         // build table to display results
61         print( "<tr>" );
62
63         foreach ( $row as $key => $value )
64             print( "<td>$value</td>" );
65
66         print( "</tr>" );
67     }
68
69     mysql_close( $database );
70
71     ?>
72 </table>
73
74 </body>
75 </html >

```

Retrieve each mailing list member record from the database.

Dynamically create a table containing each mailing list member.

PHP

122

Esecuzione



PHP

123

Precedenza degli Operatori (1)

Operator	Type	Associativity
new	constructor	none
[]	subscript	right to left
~	bitwise not	right to left
!	not	
++	increment	
--	decrement	
-	unary negative	
@	error control	
*	multiplication	left to right
/	division	
%	modulus	
+	addition	left to right
-	subtraction	
.	concatenation	
<<	bitwise shift left	left to right
>>	bitwise shift right	
<	less than	none
>	greater than	
<=	less than or equal	
>=	greater than or equal	
==	equal	none
!=	not equal	
===	identical	
!==	not identical	

Fig. 26.27 PHP operator precedence and associativity.

PHP

124

Precedenza degli Operatori (2)

Operator	Type	Associativity
&	bitwise AND	left to right
^	bitwise XOR	left to right
	bitwise OR	left to right
&&	logical AND	left to right
	logical OR	left to right
=	assignment	left to right
+=	addition assignment	
-=	subtraction assignment	
*=	multiplication assignment	
/=	division assignment	
&=	bitwise AND assignment	
=	bitwise OR assignment	
^=	bitwise exclusive OR assignment	
.=	concatenation assignment	
<<=	bitwise shift left assignment	
>>=	bitwise shift right assignment	
and	logical AND	left to right
xor	exclusive OR	left to right
or	logical OR	left to right
,	list	left to right

Fig. 26.27 PHP operator precedence and associativity.