

PHP

Ingegneria del software

Vincenzo Bonnici

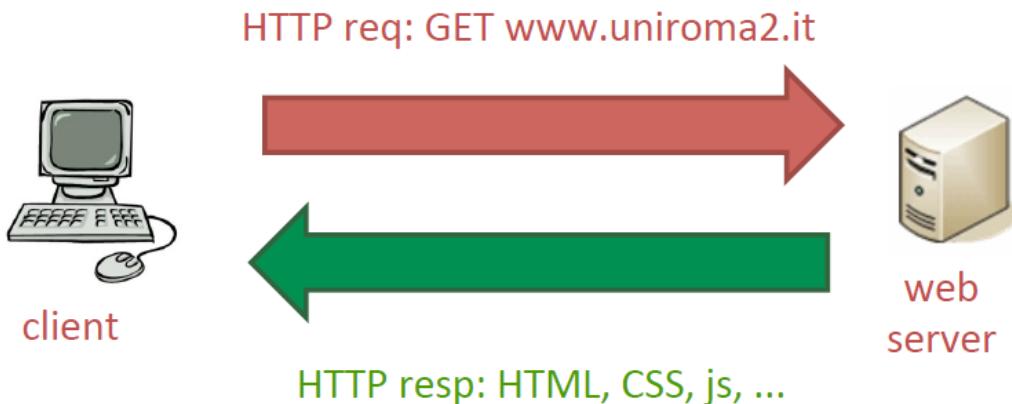
Corso di Laurea in Informatica

Dipartimento di Scienze Matematiche, Fisiche e Informatiche

Università degli Studi di Parma

2025-2026

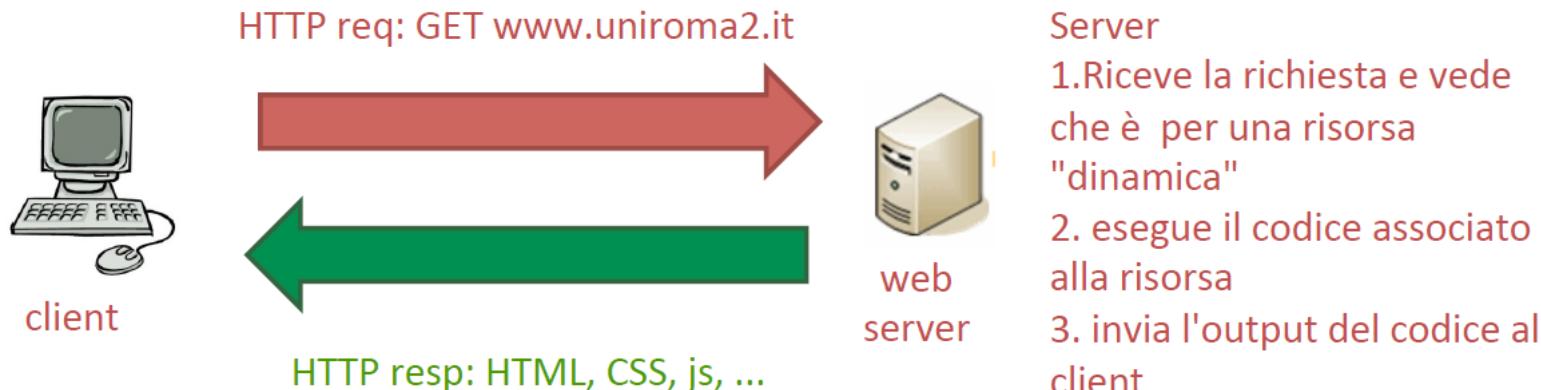
Richiesta ad un sito statico



Server

1. Riceve una richiesta per un file
2. Cerca il file e se lo trova lo invia in risposta

Richiesta ad un sito dinamico



Il server deve rispondere dinamicamente se è necessario fornire risultati diversi a seconda della situazione

- Ora o data
- richieste specifiche
- Contenuto del DB
- form o autenticazione

- **Php** – linguaggio specifico per il web, open source, interpretato
- **CGI/Perl** – Più vecchio del php ma difficile da usare. Usato per creare degli script nei sistemi unix
- **ASP.NET** – Linguaggio Microsoft, licenza commerciale; dipendente dalla piattaforma.
- **Coldfusion** – linguaggio Adobe per chi non sa programmare, facile. costoso il server.
- **Python** – Linguaggio ad oggetti per scripting generici.
- **Java (JSP o servlet)**- Java server-side programming, richiede Apache Tomcat o simili.
- **Ruby** – linguaggio generico a oggetti
- **Javascript** – server di script come node.js
... un qualsiasi linguaggio con una libreria in grado di comunicare in HTTP

- **Embedding**
 - L'HTML è inserito nel codice (con delle print) o è il codice inserito nell'HTML?
- **Flexibility**
 - quante strade ho per la soluzione del mio problema?
- **Usability**
 - quanto è difficile da imparare e da usare il linguaggio?
- **Security**
 - sviluppare nel linguaggio scelto è sicuro o devo scrivere del codice per garantire la sicurezza?
 - Ci sono buchi di sicurezza nel linguaggio?
- **Speed of execution**
 - Con che velocità viene eseguito uno script?
- **Generality**
 - Il linguaggio è generico o specifico?

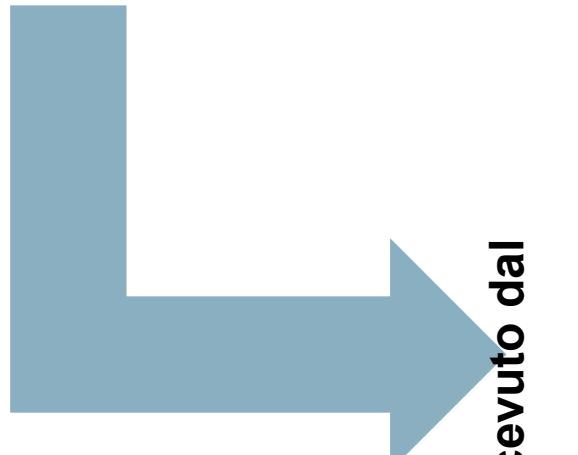
- PHP è l'**acronimo** ricorsivo di “PHP HyperText Preprocessor”
 - Inizialmente era l'acronimo di “Personal Home Page”
- PHP è un **linguaggio di programmazione** creato da Rasmus Lerdorf nel 1994 per costruire delle estensioni in documenti HTML e migliorare così la sua home page personale
 - **Porzioni del documento HTML sono generate dinamicamente**
- PHP convive normalmente all'interno di documenti HTML
 - È possibile creare degli script in PHP eseguiti da una shell
- PHP viene normalmente eseguito dal **server** prima che la pagina venga inviata all'utente
- L'output di PHP è normalmente codice HTML per il browser, ma ci sono molte eccezioni
 - Può generare anche immagini, documenti pdf, documenti XML, ..
- PHP è **open source**

- La curva di apprendimento è brevissima
- Veloci tempi di sviluppo
- Alte prestazioni e stabilità
- Supporto dei sistemi operativi principali (UNIX, Linux, Windows, ...)
- Supporto nativo per i database più popolari
- Molte librerie built-in
- Molti framework disponibili sviluppati in PHP, ad esempio CMS

```
<!DOCTYPE html>
<html>
<head>
    <title>Page Title</title>
</head>

<body>
<?php
    echo "Ciao a tutti! Oggi è ".date('l');
?>
</body>
</html>
```

File lato server



File ricevuto dal
client

```
<!DOCTYPE html>
<html>
<head>
    <title>Esempio 1</title>
</head>

<body>
Ciao a tutti! Oggi è Friday
</body>
</html>
```



1. richiesta di un documento con estensione .php
2. il server invia il documento al PHP
3. PHP interpreta lo script produce un output
4. il server produce la risposta HTTP

Esempi

Generazione di codice HTML tramite echo (print) multiple

```
<?php
    echo "<!DOCTYPE html>";
    echo "<html>";
    echo "<head>";
    echo "<title>Esempio 2</title>";
    echo "</head>";
    echo "<body>";
    echo "Ciao a tutti! Oggi è ";
    echo date('l');
    echo "</body>";
    echo "</html>";
```

Generazione di codice HTML tramite echo singola variabile

```
<?php
    $out = "<!DOCTYPE html>
              <html>
                <head>
                  <title>Esempio 2</title>
                </head>
                <body>";
    $out .= "Ciao a tutti! Oggi è ";
    $out .= date('l');
    $out .= "</body>
              </html>";

    echo $out;
```

Possiamo innestare codice PHP in qualsiasi parte del documento

```
<?php
header("Content-type: text/css");

$h1_color = '#e5e6e7';
?>

h1{
    color: <?php echo $h1_color; ?>;
}
```

<?php ... ?>

- Tutto quello che è racchiuso fra i tag <?php ... ?> viene interpretato dal modulo PHP
 - in alcuni casi si può omettere il tag di chiusura
- Metodi alternativi per inserire codice php
 - da configurare in **php.ini**
 - <? CODICE PHP ?> (short_open_tag) – **NON PIU' SUPPORTATO**
 - <% CODICE PHP %> (asp_tags)
 - <**script language="php"**> CODICE PHP </script>

- Per vedere quali particolari estensioni sono state installate o per vedere come il file php.ini è stato configurato è possibile eseguire il seguente script

```
<?php phpinfo(); ?>
```

- Inserire la riga precedente in un file (e.g., info.php) e poi invocare <http://localhost/info.php>
- La funzione phpinfo() crea una pagina HTML contenente informazioni su come PHP è stato installato

- Il server non comunica gli errori al client
- Possiamo modificare una direttiva in php.ini per mostrare nel browser gli eventuali errori (solo il primo) di script PHP
- Cercare la direttiva display_errors e settare
 - **display_errors = On**
- **SOLO PER DEBUGGING**

- Riga singola

```
// This is a comment
```

- Riga multipla

```
/* This is a section of multiline  
comments which  
will not be interpreted */
```

- Un file può contenere molto blocchi php
- Istruzioni nello stesso blocco php sono divise da ;

```
<?php  
    echo 'Questo è un test';  
?  
  
<?php echo 'Questo è un test' ?>  
  
<?php echo 'Qui è stato omesso il tag di chiusura';
```

- In alcuni casi si può omettere
 - se ho il tag di chiusura
 - se ci sono le graffe

- Nomi ammessi per
 - variabili
 - funzioni
 - costanti
 - classi
- Il primo carattere deve essere:
 - una lettera minuscola o maiuscola
 - un _ (undescore)
 - un carattere ASCII tra 0x7f e 0xFF
- I successivi possono essere anche 0–9

- Le variabili sono precedute dal simbolo \$
 - e poi l'identificatore (case sensitive)
 - alcuni nomi non sono ammessi (es. \$this)
- Non vanno dichiarate, ne inizializzate
 - basta usarle e sono create

```
<?php  
$var = 'Bob';  
$Var = 'Joe';  
echo "$var, $Var";      // outputs "Bob, Joe"  
  
$4site = 'not yet';    // invalid; starts with a number  
$_4site = 'not yet';   // valid; starts with an underscore  
$täyte = 'mansikka';  // valid; 'ä' is (Extended) ASCII 228.  
?>
```

Tipi di dato: interi

Range: da -2,147,483,648 a +2,147,483,647

– ossia -2^{31} ; $2^{31}-1$ (range dei long)

```
<?php
$a = 1234; // decimal number
$a = -123; // a negative number
$a = 0123; // octal number (equivalent to 83 decimal)
$a = 0x1A; // hexadecimal number (equivalent to 26 decimal)
$a = 0b11111111; // binary number (equivalent to 255 decimal)
?>
```

Per verificare se una variabile è un intero si può usare **is_int()** o **is_integer()**

Per vedere una variabile si usa **var_dump()**

```
$i=100;
if(is_int($i)){
    echo '$a è un intero';
    var_dump($i); //int(100)
}
```

Tipi di dato: floating point

Range positivo: da 2.2E-308 a 1.7E+308
– a 32 bit

```
<?php  
$a = 1.234;  
$b = 1.2e3;  
$c = 7E-10;  
?>
```

Per verificare se una variabile è un float si può usare `is_float()` o `is_real()`

Nei confronti attenti alle approssimazioni

```
if (intval($a * 1000) == intval($b * 1000)) {  
    // numbers equal to three decimal places  
}
```

- Sono sequenze di caratteri ad 1 byte
- Possono essere definite con apici singoli o doppi

- Differenze fra apici

- Interpretazione del testo
- Sequenze di escape
 - singolo apice solo \\ e \'
 - doppio apice quasi tutte
 - \n, \r, \t, etc.

```
$name = "Guido";
echo "Hi, $name\n";
echo 'Hi, $name';
```

Hi, Guido
Hi, \$name

- Sintassi per stringhe multiriga: <<<identificatore
 - dalla riga successiva inizia a definire la stringa
 - fino a che non trova una riga che contiene solo l'identificatore
 - e se necessario il ;
 - definisce stringhe equivalenti a quelle con "

```
$strtest=<<<PPL
<pre>ciao a tutti
questa è
una stringa
preformattata</pre>
PPL;
echo $strtest;
```

Tipi di dato: stringhe

- Sono sequenze di caratteri ad 1 byte
- Possono essere definite con apici singoli o doppi
- Differenze fra apici
 - Interpretazione del testo
 - Sequenze di escape
 - singolo apice solo \\ e \'
 - doppio apice quasi tutte
 - \n, \r, \t, etc.

```
$name = "Guido";
echo "Hi, $name\n";
echo 'Hi, $name'';

Hi, Guido
Hi, $name
```

- Sintassi per stringhe multiriga: <<<identificatore
 - dalla riga successiva inizia a definire la stringa
 - fino a che non trova una riga che contiene solo l'identificatore
 - e se necessario il ;
 - definisce stringhe equivalenti a quelle con "

```
$strtest=<<<PPL
<pre>ciao a tutti
questa è
una stringa
preformattata</pre>
PPL;
echo $strtest;
```

- Sono falso
 - The keyword `false`
 - The integer 0
 - The floating-point value 0.0
 - The empty string ("") and the string "0"
 - An array with zero elements
 - An object with no values or functions
 - The `NULL` value
- Tutto il resto è vero
- Keyword: `true`, `false`
 - case insensitive

Per verificare se una variabile è un boolean si può usare `is_bool()`

- Sono falso
 - The keyword false
 - The integer 0
 - The floating-point value 0.0
 - The empty string ("") and the string "0"
 - An array with zero elements
 - An object with no values or functions
 - The NULL value
- Tutto il resto è vero
- Keyword: true, false
 - case insensitive

```
<?php
var_dump((bool) "");           // bool(false)
var_dump((bool) 1);            // bool(true)
var_dump((bool) -2);          // bool(true)
var_dump((bool) "foo");        // bool(true)
var_dump((bool) 2.3e5);        // bool(true)
var_dump((bool) array(12));    // bool(true)
var_dump((bool) array());      // bool(false)
var_dump((bool) "false");      // bool(true)
?>
```

Per verificare se una variabile è un boolean
si può usare is_bool()

Tipi di dato: array

Gruppo di valori identificati da

- un numero (**array indicizzati**)
- o da un nome (**array associativi**)

```
$person[0] = "Edison";
$person[1] = "Wankel";
$person[2] = "Crapper";
```

```
$person[] = "Edison";
$person[] = "Wankel";
$person[] = "Crapper";
```

```
$person = array("Edison","Wankel","Crapper");
```

```
$creator['Light bulb'] = "Edison";
$creator['Rotary Engine'] = "Wankel";
$creator['Toilet'] = "Crapper";
```

```
$creator = array('Light bulb'      => "Edison",
                 'Rotary Engine' => "Wankel",
                 'Toilet'         => "Crapper");
```

```
$tar[0]="ciao";
$tar[1]=1;
$tar[3]=3.0;
$tar['ppl']=false;
var_dump($tar);
// array(4) { [0]=> string(4) "ciao" [1]=> int(1)
//           [3]=> float(3) ["ppl"]=> bool(false) }
```

Per verificare se una variabile è un array si può usare `is_array()`

- Posso accedere ad un valore dell'array mediante la chiave
- La chiave può essere inserita senza apici, con apici singoli o doppi, o mediante una variabile contenente la chiave

```
echo $tar[ppl];
echo $tar['ppl'];
echo $tar["ppl"];
$key="ppl";
echo $tar[$key];
```

Elementi non inizializzati

```
$professori[] = "loreti";
$professori[] = "bianchi";
$professori[2] = "bracciale";
$professori[10] = "detti";
printf("L'array contiene %d elementi\n", count($professori));
print("<br>");
for($i=0; $i< count($professori); $i++) {
    echo $i+1,") ";
    print(ucwords($professori[$i])); print("<br>");
}
echo "Undicesimo elemento: ",$professori[10], "<br>";
echo "Centesimo elemento: ", $professori[99];
```

L'array contiene 4 elementi

- 1) Loreti
- 2) Bianchi
- 3) Bracciale
- 4)

Undicesimo elemento: detti

Centesimo elemento:

NULL

- Rappresenta una variabile di qualunque tipo senza valore
 - non è case sensitive

```
$aleph = "beta";
$aleph = null;      // variable's value is gone
$aleph = Null;      // same
$aleph = NULL;      // same
```

Per verificare se una variabile è NULL si può usare `is_null()`

Assegnazioni

- L'operatore = assegna il valore alle variabili
 - da un literal
 - da un'altra variabile

```
$a=1;  
$b=2;  
$c = $b;  
echo "\$a=$a, \$b=$b, \$c=$c\n";  
$c = $b = $a;  
echo "\$a=$a, \$b=$b, \$c=$c\n";
```

```
$a=1, $b=2, $c=2  
$a=1, $b=1, $c=1
```

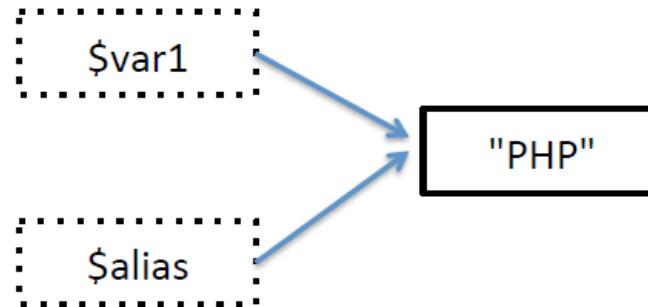
Assegnare riferimenti

- Si possono creare degli alias delle variabili

```
$var1 = "PHP";
$alias =& $var1;
$var1 .= " è bello!";
print "<pre>\$var1 vale $var1\n";
print "\$alias vale $alias\n";

$alias = "Programming $alias";
print "\$alias vale $alias\n";
print "\$var1 vale $var1</pre>";
```

```
$var1 vale PHP è bello!
$alias vale PHP è bello!
$alias vale Programming PHP è bello!
$var1 vale Programming PHP è bello!
```



```
$var1 = "Ciao";
$var2 =& $var1;
unset($var1);
print $var2; // Ciao
```

Conversioni automatiche

Type of first operand	Type of second operand	Conversion performed
Integer	Floating point	The integer is converted to a floating-point number.
Integer	String	The string is converted to a number; if the value after conversion is a floating-point number, the integer is converted to a floating-point number.
Floating point	String	The string is converted to a floating-point number.

- **Aritmetici**

+,-,*, /,%

- **Assegnamento**

= (supporta la forma abbreviata: e.g., +=)

- **Incremento/decremento**

++ e -- (prefisso/postfisso)

- **Operatori bitwise**

&, |, ^ (xor), ~, <<, >>

- **Confronto**

==, === (tipato), !=, !==, <, >, <=, >=

- **Logici**

&&, and, ||, or, xor, !

- **Condizionale**

? : (if aritmetico)

Operatori di casting

Operator	Synonymous operators	Changes type to
(int)	(integer)	Integer
(bool)	(boolean)	Boolean
(float)	(double), (real)	Floating point
(string)		String
(array)		Array
(object)		Object
(unset)		NULL

```
$a = "5";
$b = (int) $a;
```

```
$a = array('name' => "Fred", 'age' => 35, 'wife' => "Wilma");
$o = (object) $a;
echo $o->name;
```

Fred

```
class Person
{
    var $name = "Fred";
    var $age = 35;
}

$o = new Person;
$a = (array) $o;

print_r($a);

Array (
    [name] => Fred
    [age] => 35
)
```

Operatori di confronto

- L'operatore **==** testa se due espressioni sono uguali (anche dopo aver effettuato delle conversioni di Epo)
 - “1” == “1” -> true
 - “1” == 1 -> true
- L'operatore **==** testa se due espressioni sono identiche (le due espressioni devono essere uguali e dello stesso tipo)
 - “1” == “1” -> true
 - “1” == 1 -> false
- Confronti tra tipi diversi

First operand	Second operand	Comparison
Number	Number	Numeric
String that is entirely numeric	String that is entirely numeric	Numeric
String that is entirely numeric	Number	Numeric
String that is entirely numeric	String that is not entirely numeric	Numeric
String that is not entirely numeric	Number	Lexicographic
String that is not entirely numeric	String that is not entirely numeric	Lexicographic

`if(){...}`

`if(): ... endif;`

`if(){...} else{...}`

`if(): ... else: ... endif;`

`if(){...} elseif(){...} else{...}`

`if(): ... elseif(): ... else: ... endif;`

`while(){...}`

`while() ... endwhile;`

`for(){...}`

`for(): ... endfor;`

`foreach(){...}`

`foreach(): ... endforeach;`

Strutture di controllo

if

```
if ($a > $b)
    echo "a is bigger than b";
```

```
if ($a > $b) {
    echo "a is bigger than b";
    $b = $a;
}
```

if else



```
if ($a > $b) {
    echo "a is greater than b";
} else {
    echo "a is NOT greater than b";
}
```

if elseif else

```
if ($a > $b) {
    echo "a is bigger than b";
} elseif ($a == $b) {
    echo "a is equal to b";
} else {
    echo "a is smaller than b";
}
```

Strutture di controllo

if endif

```
<?php if ($a == 5): ?>  
A is equal to 5  
<?php endif; ?>
```

if elseif else endif

```
<?php  
if ($a == 5):  
    echo "a equals 5";  
    echo "...";  
elseif ($a == 6):  
    echo "a equals 6";  
    echo "!!!";  
else:  
    echo "a is neither 5 nor 6";  
endif;  
?>
```

if endif e html

```
<? if ($user_validated) :?>  
<table>  
  <tr>  
    <td>First Name:</td><td>Sophia</td>  
  </tr>  
  <tr>  
    <td>Last Name:</td><td>Lee</td>  
  </tr>  
</table>  
<? else: ?>  
  Please log in.  
<? endif ?>
```

switch

```
switch ($i) {  
    case 0:  
        echo "i equals 0";  
        break;  
    case 1:  
        echo "i equals 1";  
        break;  
    case 2:  
        echo "i equals 2";  
        break;  
}
```

switch 2

```
switch ($i) {  
    case "apple":  
        echo "i is apple";  
        break;  
    case "bar":  
        echo "i is bar";  
        break;  
    case "cake":  
        echo "i is cake";  
        break;  
}
```



while

```
$i = 1;  
while ($i <= 10) {  
    echo $i++; /* the printed value would be  
                  $i before the increment  
                  (post-increment) */  
}
```

do while

```
$i = 0;  
do {  
    echo $i;  
} while ($i > 0);
```

while endwhile

```
$i = 1;  
while ($i <= 10):  
    echo $i;  
    $i++;  
endwhile;
```

Strutture di controllo

for break

```
for ($i = 1; ; $i++) {  
    if ($i > 10) {  
        break;  
    }  
    echo $i;  
}
```

for 2

```
for($i = $x = $z = 1; $i <= 10;$i++, $x+=2, $z=$p){  
  
    $p = $i + $x;  
  
    print "\$i = $i , \$x = $x , \$z = $z \n";  
}
```

foreach

```
$a = array(1, 2, 3, 17);  
  
foreach ($a as $v) {  
    echo "Current value of $a: $v.\n";  
}
```

```
Foreach ($arr as $key => $value) {  
    echo "Key: $key; Value: $value<br />\n";  
}
```

```
$a = array(  
    "one" => 1,  
    "two" => 2,  
    "three" => 3,  
    "seventeen" => 17  
);
```

```
foreach ($a as $k => $v) {  
    echo "\$a[$k] => $v.\n";  
}
```

Iterare su array

- Ogni array in PHP conserva traccia dell'elemento con cui stiamo lavorando
 - Il puntatore all'elemento corrente è noto come **iteratore**
- La funzione **each()** restituisce un array associativo che contiene due elementi e muove l'iteratore una posizione in avanti
 - Il primo elemento è indicizzato dalla stringa **key**
 - Il secondo elemento è indicizzato dalla stringa **value**

```
<?php
$foo = array("bob", "fred", "jussi", "jouni", "egon", "marliese");
$bar = each($foo);
print_r($bar);
?>
<?php
$fruit = array('a' => 'apple', 'b' => 'banana', 'c' => 'cranberry');

reset($fruit);
while (list($key, $val) = each($fruit)) {
    echo "$key => $val\n";
}
?>
```

Array
(
 [1] => Bob
 [value] => Bob
 [0] => Robert
 [key] => Robert
)

a => apple
b => banana
c => cranberry

Funzione dell'iteratore

- **current()**
 - Restituisce il valore dell'elemento puntato dall'iteratore corrente
- **key()**
 - Restituisce la chiave dell'elemento corrente
- **reset()**
 - Muove l'iteratore al primo elemento dell'array
- **next() – prev() – end()**
 - Spostano il puntatore

```
$transport = array('foot', 'bike', 'car', 'plane');
$mode = current($transport); // $mode = 'foot';
$mode = next($transport); // $mode = 'bike';
$mode = current($transport); // $mode = 'bike';
$mode = prev($transport); // $mode = 'foot';
$mode = end($transport); // $mode = 'plane';
$mode = current($transport); // $mode = 'plane';
```

Modificare gli elementi di un array

```
$arr = array(1, 2, 3, 4);
foreach ($arr as &$value) {
    $value = $value * 2;
}
// $arr is now array(2, 4, 6, 8)
unset($value); // break the reference with the
```

```
$arr1 = array("a" => 1, "b" => 2, "c" => 3);
$arr2 = array("x" => 4, "y" => 5, "z" => 6);

foreach ($arr1 as $key => &$val) {}
foreach ($arr2 as $key => $val) {}

var_dump($arr1);
var_dump($arr2);
```

Modificare gli elementi di un array

```
$arr = array(1, 2, 3, 4);
foreach ($arr as &$value) {
    $value = $value * 2;
}
// $arr is now array(2, 4, 6, 8)
unset($value); // break the reference with the
```

```
$arr1 = array("a" => 1, "b" => 2, "c" => 3);
$arr2 = array("x" => 4, "y" => 5, "z" => 6);

foreach ($arr1 as $key => &$val) {}
foreach ($arr2 as $key => $val) {}

var_dump($arr1);
var_dump($arr2);
```

- **Sintassi** per definire una funzione in PHP è la seguente:

```
function nome_funzione([lista parametri]) {  
    implementazione  
}
```

- **Caratteristiche**

- La lista dei parametri è una sequenza di variabili separate da virgola
- Non è permesso l'overloading
 - funzioni con lo stesso nome ma parametri differenti
- Il nome delle funzioni NON è case sensitive
- È possibile usare un numero variabile di parametri
- Inoltre ad ogni parametro può essere assegnato un valore di default

- **Sintassi** per definire una funzione in PHP è la seguente:

```
function nome_funzione([lista parametri]) {  
    implementazione  
}
```

- **Caratteristiche**

- La lista dei parametri è una sequenza di variabili separate da virgola
- Non è permesso l'overloading
 - funzioni con lo stesso nome ma parametri differenti
- Il nome delle funzioni NON è case sensitive
- È possibile usare un numero variabile di parametri
- Inoltre ad ogni parametro può essere assegnato un valore di default

```
function familyName($fname) {  
    echo "$fname Refsnes.<br>";  
}
```

```
familyName("Jani");  
familyName("Hege");  
familyName("Stale");  
familyName("Kai Jim");  
familyName("Borge");
```

Visibilità variabili

```
<!DOCTYPE html>
<html lang="it">
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <?php $saluto = "Ciao a tutti"; //variabile globale
    <h1>Start page</h1>
    <?php echo ("<b>Saluto: $saluto</b>"); ?>
  </body>
</html>
```

Scope delle variabili

```
function updateCounter()  
{  
    $counter++;  
}
```

```
$counter = 10;  
updateCounter();  
  
echo $counter;
```

10

```
function updateCounter()  
{  
    global $counter;  
    $counter++;  
}
```

```
$counter = 10;  
updateCounter();  
echo $counter;
```

11

```
function updateCounter()  
{  
    $GLOBALS[counter]++;  
}
```

```
$counter = 10;  
updateCounter();  
echo $counter;
```

11

Variabili statiche

```
function updateCounter()
{
    static $counter = 0;
    $counter++;

    echo "Static counter is now {$counter}\n";
}

$counter = 10;
updateCounter();
updateCounter();

echo "Global counter is {$counter}\n";

Static counter is now 1
Static counter is now 2
Global counter is 10
```

- si definiscono con la keyword define o con quella const
 - non rispettano gli scope
 - limitate nei valori che possono contenere

```
define("CONSTANT", "Hello world.");
echo CONSTANT; // outputs "Hello world."
echo Constant; // outputs "Constant" and issues a notice.

// Works as of PHP 5.3.0
const CONSTANT = 'Hello World';

echo CONSTANT;
```

- costanti predefinite

Nome	Descrizione
<code>__LINE__</code>	Il numero della riga corrente del file.
<code>__FILE__</code>	Il percorso completo e il nome del file con i link simbolici risolti. Se utilizzato all'interno di un'inclusione, viene restituito il nome del file incluso.
<code>__DIR__</code>	La cartella del file. Se utilizzato all'interno di un'inclusione, viene restituita la cartella del file incluso. Questo è equivalente a <code>dirname(__FILE__)</code> . Il nome della cartella non ha slash a meno che non sia la directory root.
<code>__FUNCTION__</code>	Il nome della funzione.
<code>__CLASS__</code>	Il nome della classe. Il nome della classe include il namespace in cui è stata dichiarata (es. <code>Foo\Bar</code>). Notare che da PHP 5.4 <code>__CLASS__</code> funziona anche nei traits. Quando usata in un metodo trait, <code>__CLASS__</code> è il nome della classe in cui il trait è usato.
<code>__TRAIT__</code>	Il nome del trait. Il nome del trait include il namespace in cui è stato dichiarato (es. <code>Foo\Bar</code>).
<code>__METHOD__</code>	Il nome del metodo della classe.
<code>__NAMESPACE__</code>	Il nome del namespace corrente.

Costanti

- si definiscono con la keyword define o con quella const
 - non rispettano gli scope
 - limitate nei valori che possono contenere

```
define("CONSTANT", "Hello world.");
echo CONSTANT; // outputs "Hello world."
echo Constant; // outputs "Constant" and issues a notice.

// Works as of PHP 5.3.0
const CONSTANT = 'Hello World';

echo CONSTANT;
```

- costanti predefinite

Nome	Descrizione
<code>__LINE__</code>	Il numero della riga corrente del file.
<code>__FILE__</code>	Il percorso completo e il nome del file con i link simbolici risolti. Se utilizzato all'interno di un'inclusione, viene restituito il nome del file incluso.
<code>__DIR__</code>	La cartella del file. Se utilizzato all'interno di un'inclusione, viene restituita la cartella del file incluso. Questo è equivalente a <code>dirname(__FILE__)</code> . Il nome della cartella non ha slash a meno che non sia la directory root.
<code>__FUNCTION__</code>	Il nome della funzione.
<code>__CLASS__</code>	Il nome della classe. Il nome della classe include il namespace in cui è stata dichiarata (es. <code>Foo\Bar</code>). Notare che da PHP 5.4 <code>__CLASS__</code> funziona anche nei traits. Quando usata in un metodo trait, <code>__CLASS__</code> è il nome della classe in cui il trait è usato.
<code>__TRAIT__</code>	Il nome del trait. Il nome del trait include il namespace in cui è stato dichiarato (es. <code>Foo\Bar</code>).
<code>__METHOD__</code>	Il nome del metodo della classe.
<code>__NAMESPACE__</code>	Il nome del namespace corrente.

\$GLOBALS — References all variables available in global scope

\$_SERVER — Server and execution environment information

\$_GET — HTTP GET variables

\$_POST — HTTP POST variables

\$_FILES — HTTP File Upload variables

\$_REQUEST — HTTP Request variables

\$_SESSION — Session variables

\$_ENV — Environment variables

\$_COOKIE — HTTP Cookies

\$php_errormsg — The previous error message

\$HTTP_RAW_POST_DATA — Raw POST data

\$http_response_header — HTTP response

headers

\$argc — The number of arguments passed to script

\$argv — Array of arguments passed to script

```
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['REMOTE_ADDR'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
```

```
var_dump($_SERVER);
```

Variabili predefinite

```
<?php
if(isset($_GET['nome'])){
    $str="Ciao ".$_GET['nome'];
}else{
    $str="Ciao a Tutti!!";
}
?>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <h1><?php echo $str?></h1>
    </body>
</html>
```

Includere file

- **require()**

- genera un errore fatale se non si trova il file da includere

- **include()**

- genera un warning se non trova il file

```
include("header.html");
include("shared-code.php");
```

- Sopprimere gli errori

- @include**

- In genere

- require è usato per includere delle librerie
 - include è usato per includere del codice

HTML comune a più documenti

- Inclusioni di file con estensione .php includono l'output dello script

- Se vogliamo essere sicuri che il file sia inserito una sola volta usiamo **require_once** e **include_once**

```
<!DOCTYPE html>
<!-- this is header.html -->
<html><head><title>This is some common code</title>
...
HTML
```

```
# this PHP file re-uses header.html's HTML content
include("header.html");
```

Funzioni in comune

```
<?php
# this is common.php
function useful($x) { return $x * $x; }
function top() {
    ?>
    <!DOCTYPE html>
    <html><head><title>This is some common code</title>
    ...
    <?php
}
PHP
```

```
include ("common.php") ;      # this PHP file re-uses
                                # common.php's PHP code
$y = useful(42) ;              # call a shared function
top() ;                        # produce HTML output
...
```

Funzioni in comune

```
<?php
# this is common.php
function useful($x) { return $x * $x; }
function top() {
    ?>
    <!DOCTYPE html>
<html><head><title>This is some common code</title>
...
<?php
}
PHP
```

```
include("common.php");      # this PHP file re-uses
common.php's PHP code
$y = useful(42);           # call a shared function
top();                      # produce HTML output
...
```

```
<?php include "header.html"; ?>
content
<?php include "footer.html"; ?>
```

```
<?php require "design.php";
header(); ?>
content
<?php footer();
```

```
<?php
$pageTitle = "Titolo della pagina";
include ("inc/header.php");
?>
```

Html for this site

```
<?php
include ("inc/footer.php");
?>
```

Funzioni su array

Funzione	Descrizione
<u>count</u>	numero elementi
<u>print_r</u>	stampa il contenuto
<u>array_pop</u> , <u>array_push</u> , <u>array_shift</u> , <u>array_unshift</u>	usare un array come pila o coda
<u>in_array</u> , <u>array_search</u> , <u>array_reverse</u> , <u>sort</u> , <u>rsort</u> , <u>shuffle</u>	ricerca ed ordinamento
<u>array_fill</u> , <u>array_merge</u> , <u>array_intersect</u> , <u>array_diff</u> , <u>array_slice</u> , <u>range</u>	creare o riempire
<u>array_sum</u> , <u>array_product</u> , <u>array_unique</u> , <u>array_filter</u> , <u>array_reduce</u>	elaborare elementi

```
# index 0123456789012345
$name = "Pierpaolo Loreti";
$length = strlen($name);                      # 16
$cmp = strcmp($name, "Rossi");                 # != 0
$index = strpos($name, "a");                   # 5
$first = substr($name, 10, 6);                  # Loreti
$name = strtoupper($name);                     # PIERPAOLO LORETI
```

- ## Altre funzioni

- strlen, strpos, strcmp, substr
- strtolower, strtoupper
- explode, implode
- ltrim, rtrim, trim
 - Eliminano gli spazi bianchi a sinistra, a destra

Funzioni matematiche

```
$a = 3;  
$b = 4;  
$c = sqrt(pow($a, 2) + pow($b, 2))  
  
$rval = rand ( 1, 10 );
```

<u>abs</u>	<u>ceil</u>	<u>cos</u>	<u>floor</u>	<u>log</u>	<u>log10</u>	<u>max</u>
<u>min</u>	<u>pow</u>	<u>rand</u>	<u>round</u>	<u>sin</u>	<u>sqrt</u>	<u>tan</u>

funzioni

M_PI	M_E	M_LN2
------	-----	-------

costanti

Funzioni matematiche

```
$a = 3;  
$b = 4;  
$c = sqrt(pow($a, 2) + pow($b, 2))  
  
$rval = rand ( 1, 10 );
```

<u>abs</u>	<u>ceil</u>	<u>cos</u>	<u>floor</u>	<u>log</u>	<u>log10</u>	<u>max</u>
<u>min</u>	<u>pow</u>	<u>rand</u>	<u>round</u>	<u>sin</u>	<u>sqrt</u>	<u>tan</u>

funzioni

M_PI	M_E	M_LN2
------	-----	-------

costanti

Funzioni per file

NOME	CATEGORIA
<u>file</u> , <u>file_get_contents</u> , <u>file_put_contents</u>	reading/writing entire files
<u>basename</u> , <u>file_exists</u> , <u>filesize</u> , <u>fileperms</u> , <u>filemtime</u> , <u>is_dir</u> , <u>is_readable</u> , <u>is_writable</u> , <u>disk_free_space</u>	asking for information
<u>copy</u> , <u>rename</u> , <u>unlink</u> , <u>chmod</u> , <u>chgrp</u> , <u>chown</u> , <u>mkdir</u> , <u>rmdir</u>	manipulating files and directories
<u>glob</u> , <u>scandir</u>	reading directories

Lettura e scrittura file

contenuto di foo.txt	<code>file("foo.txt")</code>	<code>file_get_contents("foo.txt")</code>
Hello how r u? I'm fine	array("Hello\n", # 0 "how r u?\n", # 1 "\n", # 2 "I'm fine\n" # 3)	"Hello\n how r u?\n# a single \n# string I'm fine\n"

- **file**
 - function returns lines of a file as an array (\n at end of each)
- **file_get_contents**
 - returns entire contents of a file as a single string
- **file_put_contents**
 - writes a string into a file

Lettura e scrittura file

contenuto di foo.txt	<code>file("foo.txt")</code>	<code>file_get_contents("foo.txt")</code>
Hello how r u? I'm fine	array("Hello\n", # 0 "how r u?\n", # 1 "\n", # 2 "I'm fine\n" # 3)	"Hello\n how r u?\n# a single \n# string I'm fine\n"

- **file**
 - function returns lines of a file as an array (\n at end of each)
- **file_get_contents**
 - returns entire contents of a file as a single string
- **file_put_contents**
 - writes a string into a file

```
# display lines of file as a bulleted list
$lines = file("todolist.txt");
foreach ($lines as $line) {
    print $line;
}
PHP
```

- Ritorna un array con una stringa per riga
- Ogni riga finisce con \n
 - per levarlo usare il parametro FILE_IGNORE_NEW_LINES

```
$lines = file("todolist.txt",FILE_IGNORE_NEW_LINES);
PHP
```

file_get_contents

```
# reverse a file
$text = file_get_contents("poem.txt");
$text = strrev($text);
file_put_contents("poem.txt", $text);
```

PHP

- **file_get_contents** ritorna il file in una stringa
 - se non esiste genera un warning e la stringa è null
- **file_put_contents** scrive la stringa nel file
 - se non esiste viene creato

Appendere ad un file

```
# add a line to a file
$new_text = "P.S. ILY, GTG TTYL!~";
file_put_contents("poem.txt", $new_text, FILE_APPEND);
```

PHP

old contents	new contents
Roses are red, Violets are blue. All my base, Are belong to you.	Roses are red, Violets are blue. All my base, Are belong to you. P.S. ILY, GTG TTYL!~

Gestione dei file più complessa

- **fopen(nome_file, modalità);**
 - E' necessario assegnare il risultato di fopen() ad una variabile.
 - Se la funzione va a buon fine, PHP assegnerà al risultato un valore intero. Altrimenti un valore falso.
 - Richiesto da operazioni successive.
- Il sistema vi mette a disposizione un numero limitato di descrittori di files.
 - Appena terminate le operazioni su di un file, dovreste chiuderlo.
- Possono essere aperti in sei **modi** differenti:
 - “r” - sola lettura.
 - “r+” - lettura e scrittura.
 - “w” - solo scrittura.
 - “w+” - scrittura e lettura.
 - “a” - solo scrittura alla fine di un file.
 - “a+” - lettura e scrittura alla fine di un file.

- **fopen(nome_file, modalità);**
 - E' necessario assegnare il risultato di fopen() ad una variabile.
 - Se la funzione va a buon fine, PHP assegnerà al risultato un valore intero. Altrimenti un valore falso.
 - Richiesto da operazioni successive.
- Il sistema vi mette a disposizione un numero limitato di descrittori di files.
 - Appena terminate le operazioni su di un file, dovreste chiuderlo.
- Possono essere aperti in sei **modi** differenti:
 - “r” - sola lettura.
 - “r+” - lettura e scrittura.
 - “w” - solo scrittura.
 - “w+” - scrittura e lettura.
 - “a” - solo scrittura alla fine di un file.
 - “a+” - lettura e scrittura alla fine di un file.
- 4 tipi di connessioni che possono essere aperte:
 - HTTP, FTP, I/O standard, File System

Gestione dei file più complessa

- **fopen HTTP**
 - Cerca di aprire una connessione HTTP ad un file normalmente servito da un server WEB.
 - PHP ‘bara’ e fà credere che la richiesta arrivi da un browser che stà navigando.
 - \$fd = fopen("http://www.miosito.org/index.html/", "r");
- **fopen FTP**
 - Cerca di stabilire una connessione FTP ad un server remoto.
 - Il server remoto deve supportare il modo passivo.
 - Le operazioni possono essere o di scrittura o di lettura. (non tutte e due contemporaneamente).
 - \$fd=fopen("ftp://nomeutente:password@www.sito.org/pag.txt/", "r");
- **fopen filesystem**
 - È il modo più comune e anche di default
 - \$fd = fopen("/somewhere/location/myfile.txt", "r");
 - Ricordare che i files e le directory devono essere leggibili o scrivibili dal processo UID di PHP.
 - Se si condivide un server tra più utenti, qualunque altro utente PHP sarà in grado di leggere i files.

Gestione dei file più complessa

- **fread()** prende come argomenti un identificatore di file ed una dimensione del file in bytes.
- Se la dimensione di file non è sufficiente a leggere l'intero file, PHP si arresterà al byte specificato.
- A meno di non avere particolari motivi è meglio che PHP trovi la dimensione esatta di tutto il file, usando la funzione **filesize(nomefile)**.
- `$mystring = fread($fd, filesize($nomefile));`
- Estremamente utile!
 - Permette di trasformare qualsiasi file in stringa.
 - Le stringhe possono essere manipolate con milioni di funzioni!
 - Dividere le stringhe in array (`file()` , **explode()**)
- Se si vuole leggere l'output su pagina web, utilizzare la funzione `readfile(nomefile)`.

Gestione dei file più complessa

- Per fare operazioni sulle singole righe si può usare la funzione:
 - **fgets**(descrittore, bytes).

```
<?
    $fd = fopen("esempio.txt", "r");
    while(!feof($fd)) {
        $riga = fgets($fd, 4096);
        if(strcmp($riga, $destinazione) == 0)
            echo "trovata corrispondenza!";
    }
    fclose(fd); //ricordarsi di chiudere!!!
?>
```

- Se si desidera leggere il file come un array, è possibile utilizzare la funzione: **file(nomefile)**.
 - Crea un array: ogni elemento è una riga del file originale con un carattere di interruzione riga.
 - La funzione file non richiede un'operazione separata di apertura e chiusura file.

```
<?
    $fd = fopen("esempio.txt", "r"); // NON SERVE
    $myarray = file("esempio.txt");
    fclose(fd); // NON SERVE
?>
```

Gestione dei file più complessa

- **fwrite(descrittore, stringa).**
 - Prende in input un descrittore ed una stringa.
 - Restituisce il numero di caratteri scritti
- **fputs(descrittore, stringa).**
 - E' identica ad fwrite.
 - La scrittura di files in un server WEB può essere un problema di sicurezza.
 - A meno di non avere buoni motivi per farlo, non fateo!!!

```
<?
    $fd = fopen("esempio.txt", "w");
    $mystring = "Hello World!";
    $fout = fwrite($fd, $mystring);
    echo "Ho scritto $fout caratteri!";
    fclose(fd);
?>
```

Gestione dei file più complessa

- La funzione **feof** controlla la fine del file su un puntatore di file.
 - Prende come argomento un nome di file.

```
<?
    $fd = fopen("esempio.txt", "r");
    while(!feof($fd)) {
        $riga = fgets($fd, 4096);
        if(strcmp($riga, $destinazione) == 0)
            echo "trovata corrispondenza!";
    }
    fclose(fd); //ricordarsi di chiudere!!!
?>
```

- La funzione **file_exists()** può essere utilizzata quando si agisce sul file system.
- Controlla il fs locale per cercare un file dal nome specificato:

```
<?
    if(!file_exists("esempio.txt")) {
        $fd = fopen("esempio.txt", "w+");
    }
    else {
        $fd = fopen("esempio.txt", "a+");
    }
    fclose(fd); //ricordarsi di chiudere!!!
?>
```

Leggere directory

funzione	descrizione
<u>glob</u>	returns an array of all file names that match a given pattern (returns a file path and name, such as "foo/bar/myfile.txt")
<u>scandir</u>	returns an array of all file names in a given directory (returns just the file names, such as "myfile.txt")

glob

```
# reverse all poems in the poetry directory
$poems = glob("poetry/poem*.dat");
foreach ($poems as $poemfile) {
    $text = file_get_contents($poemfile);
    file_put_contents($poemfile, strrev($text));
    print "I just reversed ".
basename($poemfile) . "\n";
}
PHP
```

- **glob** supporta path con "wildcard" mediante il carattere *
- **glob("foo/bar/*.doc")**
 - returns all .doc files in the foo/bar subdirectory
- **glob("food")**
 - returns all files whose names begin with "food"
- the **basename** function strips any leading directory from a file path
 - **basename("foo/bar/baz.txt")** returns "baz.txt"

scandir

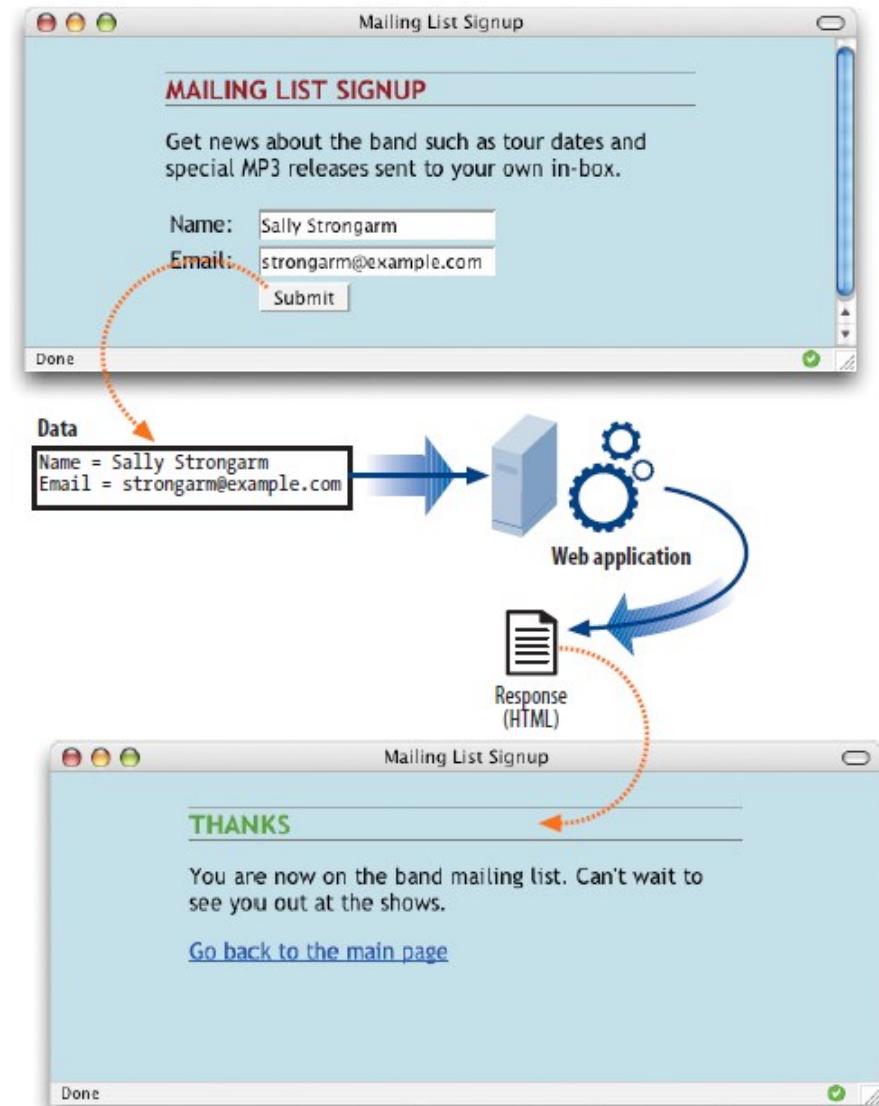
```
<ul>
  <?php foreach (scandir("taxes/old") as $filename) { ?>
  <li>I found a file: <?= $filename ?></li>
  <?php } ?>
</ul>
PHP
```

```
.
..
2007_w2.pdf
2006_1099.doc
output
```

- include la directory corrente (".") ed il parent ("..")
- non serve basename; il nome del file non contiene il path completo

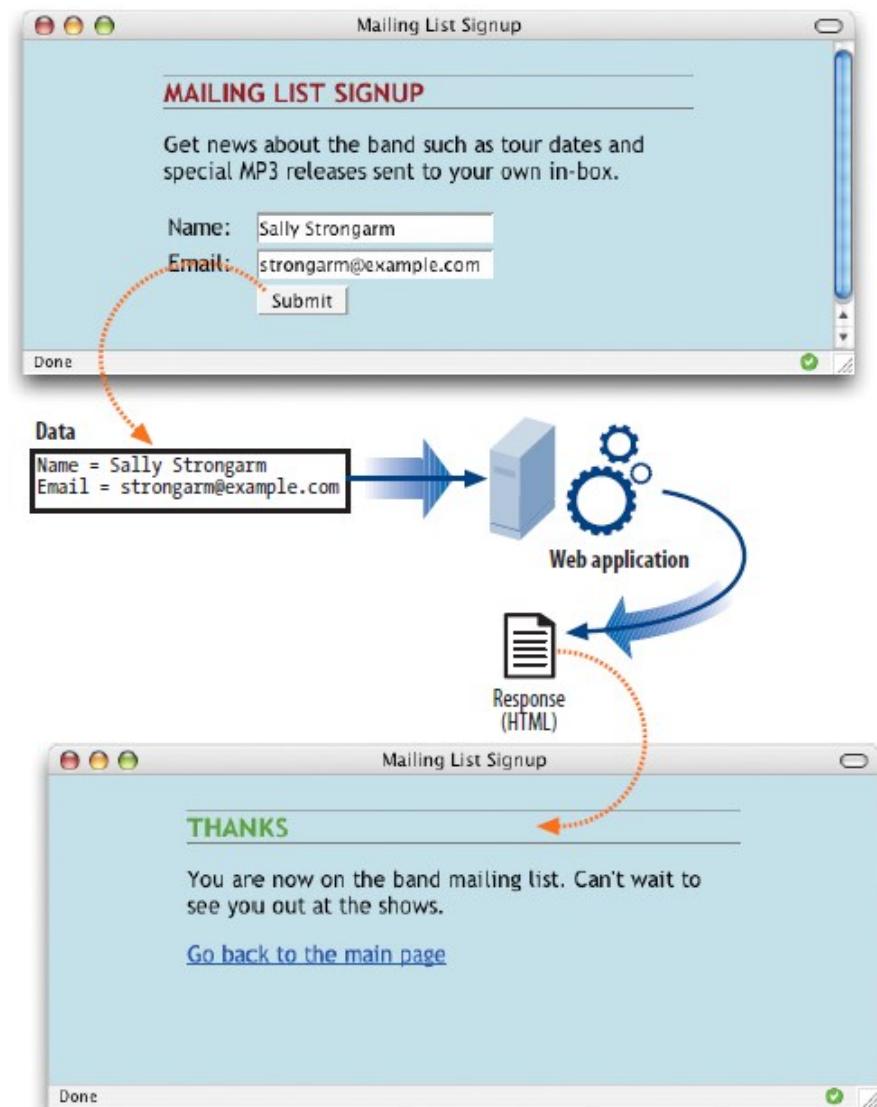
Form

- Un insieme di elementi in una pagina web con cui l'utente interagisce per inviare informazioni ad uno script
- Realizzazione di due cose
 - la pagina contiene il form
 - Lo script che riceve ed elabora le informazioni



Flusso delle operazioni

1. Apertura pagina con il form
2. Compilazione del form e click di un bottone
3. Invio delle informazioni al server
4. Uno script prende le informazioni e le processa
5. Lo script genera una pagina di risposta a seconda delle informazioni
6. Il server invia la pagina al browser che la visualizza



Elemento form

- <form>...</form>
- Serve a contenere gli elementi appartenenti al form
 - Attivi: campi di testo, bottoni, menu
 - Altri elementi descrittivi: tutti tranne form

```
<!DOCTYPE html> <html>
<head>
    <title>Mailing List Signup</title>
    <meta charset="utf-8">
</head>
<body>
<h1>Mailing List Signup</h1>
<form action="mailinglist.php" method="post">
    <fieldset>
        <legend>Join our email list</legend>
        <p>Get news about the band such as tour dates
            and special MP3 releases sent to your own in-box.</p>
        <ol>
            <li>
                <label for="firstlast">Name:</label>
                <input type="text" name="username" id="firstlast">
            </li>
            <li>
                <label for="email">Email:</label>
                <input type="text" name="email" id="email">
            </li>
        </ol>
        <input type="submit" value="Submit">
    </fieldset>
</form>
</body>
</html>
```

- I moduli rendono le pagine WEB interattive
 - L'utente interagisce con una pagina che contiene un(a) form
 - L'utente inserisce dati e invia premendo un pulsante
 - Il server riceve i dati tramite un messaggio di richiesta HTTP con metodo GET o POST (POST piu' robusto, preferibile se il server genera effetti collaterali)
 - Il server elabora i dati ricevuti ed invia all'utente la pagina contenente la risposta
-
- <form id="bollo" action="bollo.php" method="POST">
...
</form>
 - **id**: codice identificativo del modulo
 - **action**: pagina che elabora i dati
 - **method**: metodo HTTP
 - **enctype**: tipo di codifica dei dati
 - "text/plain"
 - "application/x-www-form-urlencoded" - con POST
 - "multipart/form-data" - obbligatorio se si inviano file

- **GET (default)**

- HTTP GET con i dati concatenati nella URL
- si usa ? all'inizio e & fra i dati
- `http://www.example.com/mailinglist.php?name=Sally%20Strongarm&email=strongarm%40example.com`

Dati:

`username = Sally Strongarm`
`email = strongarm@example.com`

`?name=Sally%20Strongar m&email=strongarm%40example.com`

- **accept-charset**

- Specifies the charset used in the submitted form (default: the page charset).

- **autocomplete**

- Specifies if the browser should autocomplete the form (default: on).

- **novalidate**

- Specifies that the browser should not validate the form.

- **target**

- Specifies the target of the address in the action attribute (default: _self).

Elemento form (modulo): variabili e contenuto

- Può contenere molti elementi attivi
- Ogni **elemento** definisce una **variabile** che ha un **nome**
- L'utente interagendo con il form varia il contenuto delle variabili che vengono inviate al server
- Nell'esempio
 - variabili “username” and “email”
 - “Sally Strongarm” and “strongarm@example.com” valori
 - contenuti nelle variabili
- L'attributo **name** definisce il nome della variabile

```
<textarea name="comment" rows="4" cols="45" placeholder="Leave us a comment."></textarea>
```

 - Se scrivo: “This is the best band ever!”
 - La variabile vale: comment=This%20is%20the%20best%20band%20ever%21
- I nomi delle variabili HTML li uso in php per prendere il valore dei parametri della chiamata HTTP
 - Il php se trova cose strane li converte

Elemento form (modulo): variabili e contenuto

- L'elemento INPUT definisce un campo di input.
- <input type="..." name="..." value="...">
 - **type** definisce il tipo di controllo
 - type="text" Campo di testo di una riga
 - type="password" Come TEXT, ma con l'input mascherato (*****)
 - type="hidden" non visualizzato ma inviato al server
 - type="file" selezione di file
 - type="checkbox" casella di controllo. selezionata o non selezionata
 - type="radio" casella per la scelta di una tra varie alternative.
 - type="submit" pulsante per l'invio dei contenuti del modulo
 - type="reset" pulsante per ripristinare i contenuti di un modulo ai valori di default
 - type="image" utilizzare immagini come bottoni (attributi SRC ed ALT)
 - HTML5 aggiunge: color, date, datelme, datelme-local, email, month, number, range, search, tel, lme, url, week
 - **name** nome del controllo / della variabile associata ad esso
 - **value** valore iniziale del controllo (o etichetta)
 - Valore se type in {text, password, file, hidden}
 - Etichetta se type in {submit, reset, button}
- **Altri attributi** validi solo per alcuni valori di type
 - size, checked, maxlength, src, min, max, **pattern** (verifica secondo una regex), **required**
- **Ulteriori attributi** di INPUT; valgono anche per altri controlli
 - tabindex indice di tabulazione
 - accesskey scorciatoia da tastiera
 - readonly sola lettura, valore inviato al server
 - disabled disabilitato, valore non inviato al server

INPUT TYPE="TEXT"

- MAXLENGTH Massimo numero di caratteri inseribili
- SIZE Dimensione della casella
- VALUE Valore predefinito
- Esempio

```
<input type="text" name="domanda" maxlength="25" size="5" value="Inserire qui la domanda">
```

INPUT TYPE="PASSWORD"

- Caratteri inseriti mascherati con asterischi
- Non visualizzato a video ma inviato *"in chiaro"*
- Attributi
 - MAXLENGTH
 - SIZE
 - VALUE

Tipi di controlli - INPUT

INPUT TYPE="CHECKBOX"

- Casella di controllo
- VALUE Valore inviato allo script se (e solo se) la casella è selezionata
- CHECKED Pre-seleziona la casella
- Esempio

```
<input type="checkbox" name="bianco" value="si"> bianco <br>
<input type="checkbox" name="rosso" value="si" checked="checked">
    rosso<br>
<input type="checkbox" name="verde" value="si"> verde <br>
```

INPUT TYPE="RADIO"

- Pulsanti di opzione
- Scelta esclusiva: pu. essere selezionata soltanto una delle opzioni nello stesso gruppo
- Opzioni raggruppate usando stesso nome ma valori diversi
- Scelta non obbligatoria: se non si seleziona nulla, nulla viene inviato allo script
- Esempio

HTML <input type="radio" name="argomento" value="html" />
CSS <input type="radio" name="argomento" value="css" />

Tipi di controlli - INPUT

```
<p>How old are you?</p> <ol>
  <li><input type="radio" name="age" value="under24" checked> under 24</li>
  <li><input type="radio" name="age" value="25-34"> 25 to 34</li>
  <li><input type="radio" name="age" value="35-44"> 35 to 44</li>
  <li><input type="radio" name="age" value="over45"> 45+</li>
</ol>

<p>What type of music do you listen to?</p> <ul>
<li><input type="checkbox" name="genre" value="punk" checked> Punk rock</li>
<li><input type="checkbox" name="genre" value="indie" checked> Indie rock</li>
<li><input type="checkbox" name="genre" value="hiphop"> Hip Hop</li>
<li><input type="checkbox" name="genre" value="rockabilly"> Rockabilly</li>
</ul>
```

Radio buttons	Checkbox buttons
How old are you?	What type of music do you listen to?
<input checked="" type="radio"/> under 24 <input type="radio"/> 25 to 34 <input type="radio"/> 35 to 44 <input type="radio"/> 45+	<input checked="" type="checkbox"/> Punk rock <input checked="" type="checkbox"/> Indie rock <input type="checkbox"/> Hip Hop <input type="checkbox"/> Rockabilly

Tipi di controlli - TEXTAREA

- Inserimento testo su più righe.
 - NAME: nome del controllo
 - ROWS: numero di righe
 - COLS: numero di colonne
- Esempio

```
<textarea name="corpo" rows="6" cols="35">
Testo predefinito.
Ciao come stai?
</textarea>
```

<select> ... </select>

- Lista di selezione o menù a discesa
 - Elementi della lista specificati con OPTION
- NAME
- MULTIPLE (attributo booleano)
 - Selezione di molteplici opzioni
- SIZE
 - Numero di opzioni mostrate nel controllo

OPTION

- Elemento della lista del tag SELECT
- Attributi principali:
 - VALUE
 - Indica il valore da inviare al server.
- SELECTED
 - Preselezione di una voce del menù (**selected="selected"**)
 - Anche su più elementi di tipo option se multiple="multiple"

Tipi di controlli - SELECT

```
<p>What is your favorite 80s band?  
<select name="EightiesFave">  
  <option>The Cure</option>  
  <option>Cocteau Twins</option>  
  <option>Tears for Fears</option>  
  <option>Thompson Twins</option>  
  <option value="EBTG">Everything But the Girl</option>  
  <option>Depeche Mode</option>  
  <option>The Smiths</option>  
  <option>New Order</option>  
</select>  
</p>
```

What is your favorite 80s band? 

Tipi di controlli

Raggruppare opzioni

```
<optgroup label="Argomenti lezione">
    <option value="html"> HTML </option>
    <option value="css"> CSS </option>
</optgroup>
```

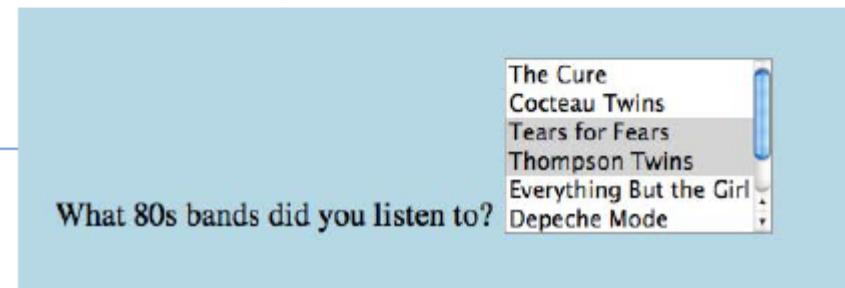
Raggruppare elementi

```
<form action="...">
    <fieldset>
        <legend>Anagrafica</legend>
        <input type="text" name="nome">
        <input type="text" name="cognome">
        <input type="text" name="eta">
    </fieldset>
</form>
```

Tipi di controlli

Tor V

```
<p>What 80s bands did you listen to?
<select name="EightiesBands" size="6" multiple>
    <option>The Cure</option>
    <option>Cocteau Twins</option>
    <option selected>Tears for Fears</option>
    <option selected>Thompson Twins</option>
    <option value="EBTG">Everything But the Girl</option>
    <option>Depeche Mode</option>
    <option>The Smiths</option>
    <option>New Order</option>
</select>
</p>
```



```
<select name="icecream" size="7" multiple>
    <optgroup label="traditional">
        <option>vanilla</option>
        <option>chocolate</option>
    </optgroup>
    <optgroup label="fancy">
        <option>Super praline</option>
        <option>Nut surprise</option>
        <option>Candy corn</option>
    </optgroup>
</select>
```

traditional
vanilla
chocolate
fancy
Super praline
Nut surprise
Candy corn

Tipi di controlli

```
<fieldset>
  <legend>Mailing List Sign-up</legend>
  <ul>
    <li><label>Add me to your mailing list
      <input type="radio" name="list" value="yes" checked="checked"></label></li>
    <li><label>No thanks <input type="radio" name="list" value="no"></label></li>
  </ul>
</fieldset>

<fieldset>
  <legend>Customer Information</legend>
  <ol>
    <li><label>Full name: <input type="text" name="username"></label></li>
    <li><label>Email: <input type="text" name="email"></label></li>
    <li><label>State: <input type="text" name="state"></label></li>
  </ol>
</fieldset>
```

Tipi di controlli - INPUT

INPUT TYPE="FILE"

- Selezione di un file locale per inviarlo al server con il modulo
- Il browser aprirà una finestra di dialogo
- Servono
 - method="post"
 - enctype="multipart/form-data"

```
<form action="/client.php" method="POST" enctype="multipart/form-data">
    <label>Send a photo to be used as your online icon <em>(optional)</em><br>
        <input type="file" name="photo" size="28"><label>
    </form>
```

File input (Firefox)

Send a photo to be used as your online icon (*optional*):

Browse...

File input (Chrome)

Send a photo to be used as your online icon (*optional*):

No file chosen

Tipi di controlli - INPUT

INPUT TYPE="HIDDEN"

- Campo nascosto a video ma inviato al server
- Usato per passare informazioni allo script sul server senza “appesantire” visivamente il form

```
<input type="hidden" name="success-link"  
      value="http://www.example.com/ littlechair_thankyou.html">
```

INPUT TYPE="SUBMIT" e "RESET"

```
<input type="submit" value="invia">  
<input type="reset" value="cancella">
```

Etichette dei controlli

Elemento label associa un'etichetta ai controllo in maniera semantica (quindi maggiormente accessibile) invece che esclusivamente in maniera visuale (p.e. tabelle)

<label>Cognome:

```
<input type="text" name="cognome">
```

```
</label>
```

<label for="cogn">Cognome:</label>

```
<input type="text" id="cogn" name="cognome">
```

– associazione implicita

```
<ul>
  <li><label><input type="checkbox" name="genre" value="punk">Punk rock</label></li>
  <li><label><input type="checkbox" name="genre" value="indie">Indie rock</label></li>
  <li><label><input type="checkbox" name="genre" value="hiphop">Hip Hop</label></li>
  <li><label><input type="checkbox" name="genre" value="rockabilly">Rockabilly</label></li>
</ul>
```

– associazione esplicita

```
<label for="form-login-username">Login account</label>
<input type="text" name="login" id="form-login-username">

<label for="form-login-password">Password</label>
<input type="password" name="password" id="form-login-password">
```

- **\$_SERVER**

- Contains useful information about the web server

- **\$_GET**

- Contains any parameters that are part of a GET request, where the keys of the array are the names of the form parameters

- **\$_POST**

- Contains any parameters that are part of a POST request, where the keys of the array are the names of the form parameters

- **\$_FILES**

- Contains information about any uploaded files

PHP: passare dati POST

```
<html>
<head>
    <title>Chunkify Form</title>
</head>
<body>
    <form action="ch.php" method="POST">
        Enter a word: <input type="text" name="word" /><br />
        How long should the chunks be?
        <input type="text" name="number" /><br />
        <input type="submit" value="Chunkify!">
    </form>
</body>
</html>
```

```
<?php

$word = $_POST['word'];
$number = $_POST['number'];

$chunks = ceil(strlen($word) / $number);
echo "The {$number}-letter chunks of '{$word}' are:<br />\n";

for ($i = 0; $i < $chunks; $i++) {
    $chunk = substr($word, $i * $number, $number);
    printf("%d: %s<br />\n", $i + 1, $chunk);
}
```

PHP: passare dati GET

```
<html>
<head>
    <title>Chunkify Form</title>
</head>
<body>
    <form action="ch.php" method="GET">
        Enter a word: <input type="text" name="word" /><br />
        How long should the chunks be?
        <input type="text" name="number" /><br />
        <input type="submit" value="Chunkify!">
    </form>
</body>
</html>
```

```
<?php

$word = $_GET['word'];
$number = $_GET['number'];

$chunks = ceil(strlen($word) / $number);
echo "The {$number}-letter chunks of '{$word}' are:<br />\n";

for ($i = 0; $i < $chunks; $i++) {
    $chunk = substr($word, $i * $number, $number);
    printf("%d: %s<br />\n", $i + 1, $chunk);
}
```

```
if ($_SERVER['REQUEST_METHOD'] == 'GET') {
    // handle a GET request
}
else {
    die("You may only GET this page.");
}
```

PHP: pagine self processing

Pagine che generano il form e che lo elaborano (DA EVITARE!!!!)

```
<html>
<head><title>Temperature Conversion</title></head>
<body>
<?php
    if ($_SERVER['REQUEST_METHOD'] == 'GET') {
?>
    <form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="POST">
        Fahrenheit temperature:
        <input type="text" name="fahrenheit" /><br />
        <input type="submit" value="Convert to Celsius!" />
    </form>
<?php
    } elseif ($_SERVER['REQUEST_METHOD'] == 'POST') {
        $fahrenheit = $_POST['fahrenheit'];
        $celsius = ($fahrenheit - 32) * 5 / 9;
        printf("%.2fF is %.2fC", $fahrenheit, $celsius);
    } else {
        die("This script only works with GET and POST requests.");
    } ?>
</body>
</html>
```

PHP: sticky form

Uso dei valori inviati per un form nella pagina di output

```
<html>
<head><title>Temperature Conversion</title></head>
<body>
    <?php $fahrenheit = $_GET['fahrenheit']; ?>
    <form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="GET">
        Fahrenheit temperature:
        <input type="text" name="fahrenheit" value="<?php echo $fahrenheit; ?>" />
        <br />
        <input type="submit" value="Convert to Celsius!" />
    </form>
    <?php if (!is_null($fahrenheit)) {
        $celsius = ($fahrenheit - 32) * 5 / 9;
        printf("%.2fF is %.2fC", $fahrenheit, $celsius);
    } ?>
</body>
</html>
```

Parametri con valori multipli

- Uso le parentesi quadre [] nel attributo name per dire che sarà un array

```
<select name="languages[]>
  <option name="c">C</input>
  <option name="c++">C++</input>
  <option name="php">PHP</input>
  <option name="perl">Perl</input>
</select>
```

```
<html>
<head><title>Personality</title></head>
<body>

<form action=<?php echo $_SERVER['PHP_SELF']; ?>" method="GET">
  Select your personality attributes:<br />
  <select name="attributes[]" multiple>
    <option value="perky">Perky</option>
    <option value="morose">Morose</option>
    <option value="thinking">Thinking</option>
    <option value="feeling">Feeling</option>
    <option value="thrifty">Spend-thrift</option>
    <option value="shopper">Shopper</option>
  </select><br />
  <input type="submit" name="s" value="Record my personality!" />
</form>

<?php if (array_key_exists('s', $GET)) {
  $description = join(' ', $GET['attributes']);
  echo "You have a {$description} personality.";
} ?>
```

PHP: sticky form

```
<html>
    <head><title>Personality</title></head>
<body>
<?php
    // fetch form values, if any
    $attrs = $_GET['attributes'];
    if (!is_array($attrs)) {
        $attrs = array();
    }

    // create HTML for identically named checkboxes
    function makeCheckboxes($name, $query, $options) {
        foreach ($options as $value => $label) {
            $checked = in_array($value, $query) ? "checked" : '';
            echo "<input type=\"checkbox\" name=\"$name\" value=\"$value\" $checked />";
            echo "$label<br />\n";
        }
    }

    // the list of values and labels for the checkboxes
    $personalityAttributes = array( 'perky'=> "Perky", 'morose' => "Morose",
                                    'thinking'=> "Thinking", 'feeling' => "Feeling",
                                    'thrifty' => "Spend-thrift", 'prodigal' => "Shopper");
?>

<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="GET">
    Select your personality attributes:<br />
    <?php makeCheckboxes('attributes[]', $attrs, $personalityAttributes); ?><br />
    <input type="submit" name="s" value="Record my personality!" /> </form>
    <?php if (array_key_exists('s', $_GET)) {
        $description = join (' ', $_GET['attributes']);
        echo "You have a {$description} personality.";
    }
?>
</body></html>
```

File upload

- Il file inviato dal form è posto fra i file temporanei con un nome casuale
- La variabile `$_FILE` contiene le informazioni necessarie per accedervi

```
<form enctype="multipart/form-data"
    action="php echo $_SERVER['PHP_SELF']; ?" method="POST">
<input type="hidden" name="MAX_FILE_SIZE" value="10240">
File name: <input name="toProcess" type="file" />
<input type="submit" value="Upload" />
</form>

if (is_uploaded_file($_FILES['toProcess']['tmp_name'])) {
    // successfully uploaded
}

move_uploaded_file($_FILES['toProcess']['tmp_name'], "path/to/put/file/{$file}");
```

File upload

```
<form enctype="multipart/form-data" action="__URL__" method="POST">
    <!-- MAX_FILE_SIZE deve precedere campo di input del nome file -->
    <input type="hidden" name="MAX_FILE_SIZE" value="30000" />
    <!-- Il nome dell'elemento di input determina il nome nell'array $_FILES -->
    Send this file: <input name="userfile" type="file" />
    <input type="submit" value="Send File" />
</form>

$uploadaddir = '/var/www/uploads/';
$uploadfile = $uploadaddir . basename($_FILES['userfile']['name']);

if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
    echo "File is valid, and was successfully uploaded.\n";
} else {
    echo "Possible attack through file upload!\n";
}
```

File upload

```
$uploaddir = '/var/www/uploads/';
$uploadfile = $uploaddir . basename($_FILES['userfile']['name']);

echo '<pre>';
if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
    echo "File is valid, and was successfully uploaded.\n";
} else {
    echo "Possibile attacco tramite file upload!\n";
}

echo 'Alcune informazioni di debug:';
print_r($_FILES);

print "</pre>";
```

Protocolli per il web

Ingegneria del software

Vincenzo Bonnici

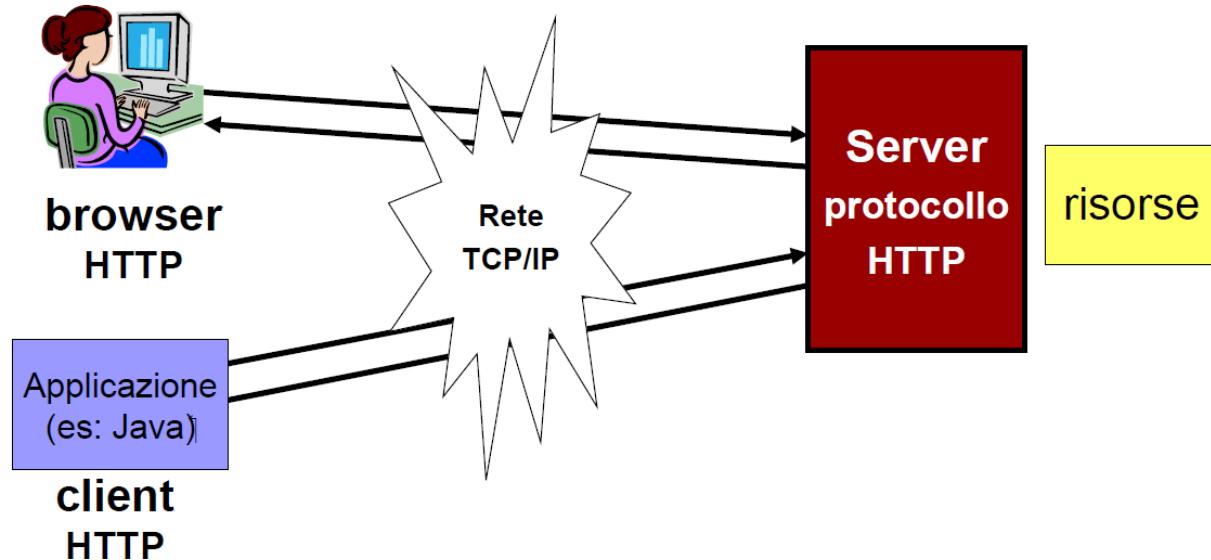
Corso di Laurea in Informatica

Dipartimento di Scienze Matematiche, Fisiche e Informatiche

Università degli Studi di Parma

2024-2025

- La storia inizia nel 1989
 - Tim Berners-Lee al CERN, progetto **WWW**
 - primi prototipi nel 1991
- Idea generale: “**distributed hypermedia**”
 - un ipertesto multimediale distribuito
 - **ipertesto** = testo a sviluppo non lineare
 - **multimediale** = più di un “medium”
 - **distribuito** = dislocato sui nodi di una rete
- Tre idee fondamentali
 - un protocollo **client-server** (HTTP)
 - un sistema di **indirizzamento** (URL)
 - un **linguaggio** per la compilazione dei doc. (HTML)
- **Inizialmente**
 - pubblicazione di contenuti (documenti)
- **Oggi**
 - La quasi totalità dei flussi Internet è basata su HTTP (testo/audio/video/file)



- Compiti del **browser**
 - visualizzare le risorse e gestire l'interazione con l'utente
- Compiti del **server**
 - nel caso di **documenti**: inviare il contenuto di file contenuti sul disco locale
 - nel caso di **servizi** interattivi: eseguire un'applicazione e restituire i risultati dell'esecuzione

“Web Characterization Terminology”, **W3C** :World Wide Web Consortium
1994: MIT + CERN + ERCIM(1995)

Risorsa : qualsiasi cosa sia accessibile su un server

Server : ruolo svolto da un'applicazione che fornisce risorse

Client : ruolo svolto da un'applicazione che richiede risorse

Messaggio: unità di comunicazione scambiata tra client e server

Richiesta : messaggio inviato dal client al server per richiedere una operazione su una risorsa

Risposta : messaggio contenente il risultato dell'esecuzione di una richiesta

Utente : soggetto che interagisce con il client per accedere risorse

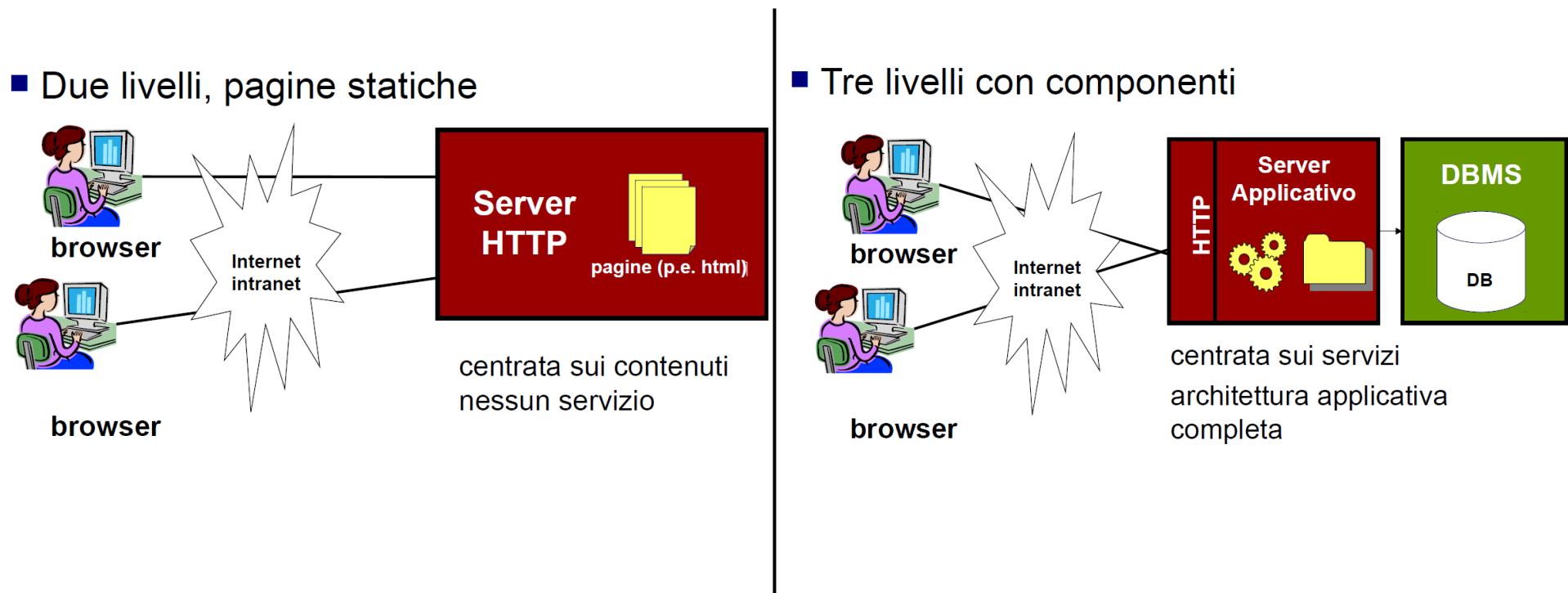
Autore o Sviluppatore : chi pubblica risorse su un server

World Wide Web

- servizio applicativo di Internet basato su un'architettura client/server
- ma non è l'unico (e.g. SMTP, POP, IMAP, FTP, SSH,...)

Nel caso del Web

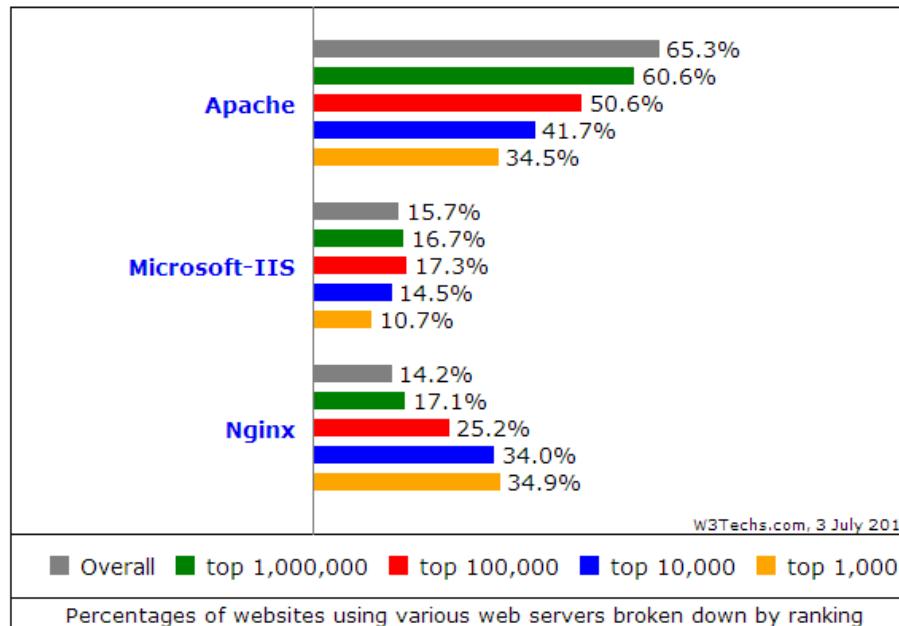
- **risorse**: documenti, immagini ed altri media, servizi interattivi
- **client**: tipicamente il browser
- **server**: server per l'accesso a risorse Web



Server web

- **Apache HTTP Server** (httpd.apache.org)
 - open source (deriva da NCSA)
 - server HTTP
 - vari protocolli di interfaccia con server applicativi (es: CGI, PHP)
- **Nginx**
 - nato per performare meglio di Apache, ha delle limitazioni in termini di flessibilità della configurazione e accesso alle risorse
- Microsoft Internet Information Services (**IIS**)
 - server HTTP
 - integrato con il server applicativo .NET
- Apache **Tomcat** (jakarta.apache.org)
 - server applicativo open source per Java Servlet e JSP
 - include server HTTP
- IBM **WebSphere**, Bea WebLogic ed altri
 - server applicativi commerciali per J2EE
 - includono server HTTP
- **JBoss** (www.jboss.org)
 - server applicativo open source per J2EE
 - include server HTTP

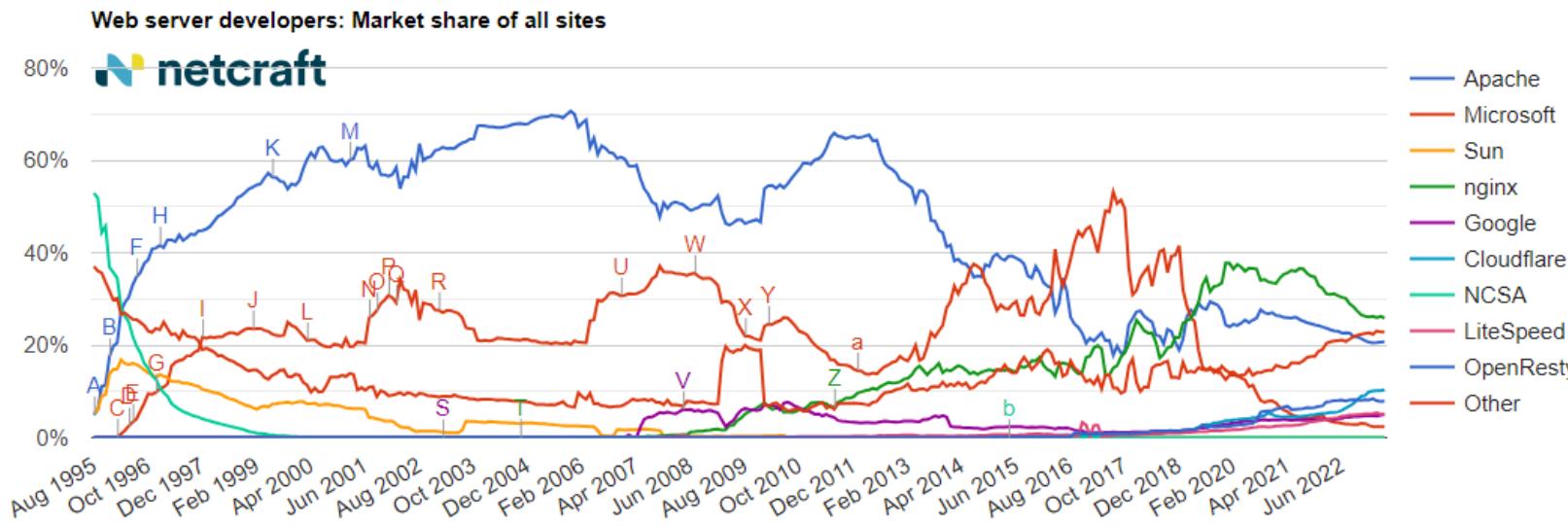
Server web



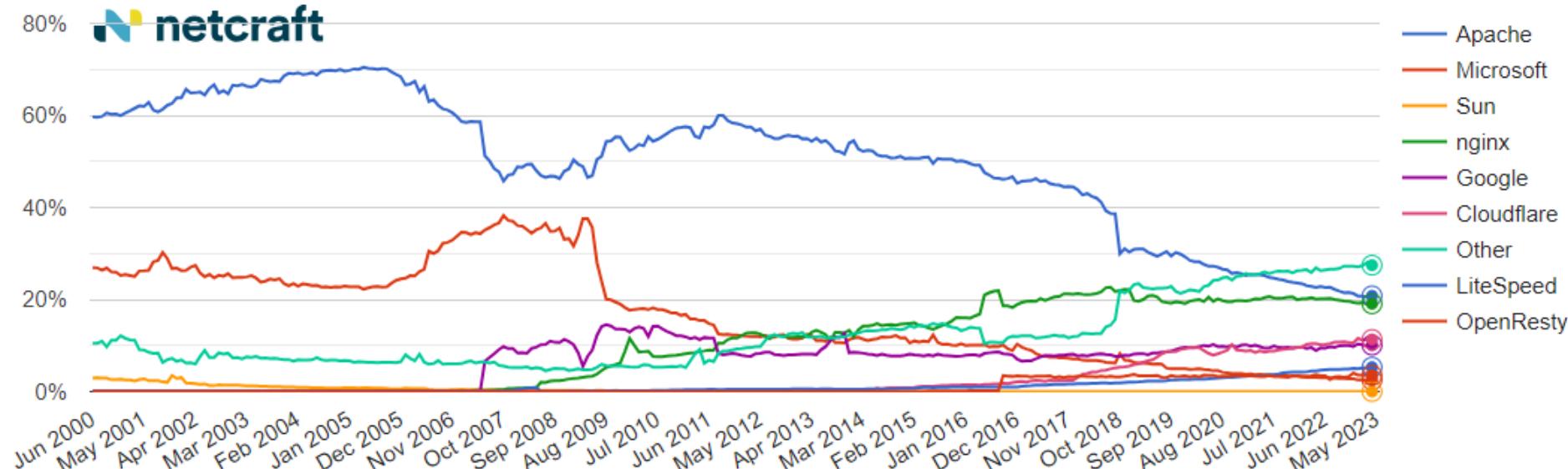
95.4% of Nginx sites are using **PHP**, much more than average. But **Nginx** is also very popular among Ruby users, with a 44.4% usage rate among sites using that language. 98.7% of all Nginx installations run on a **Unix-like** operating system, with Debian, Ubuntu and Gentoo being particularly popular choices.

https://w3techs.com/blog/entry/nginx_just_became_the_most_used_web_server_among_the_top_1000_websites

Server web



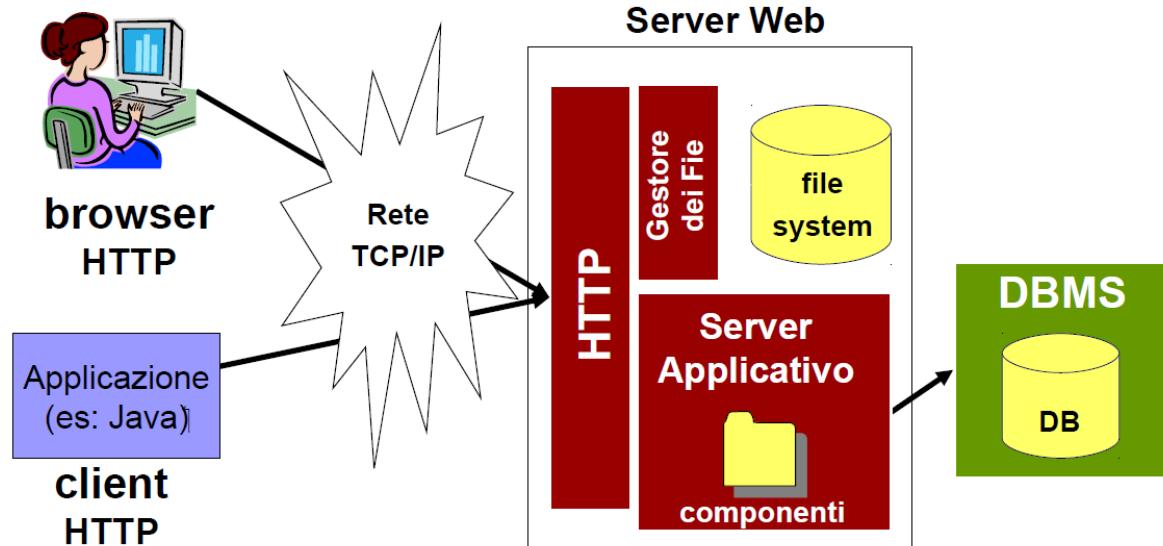
Web server developers: Market share of active sites



Architettura di un Server Web

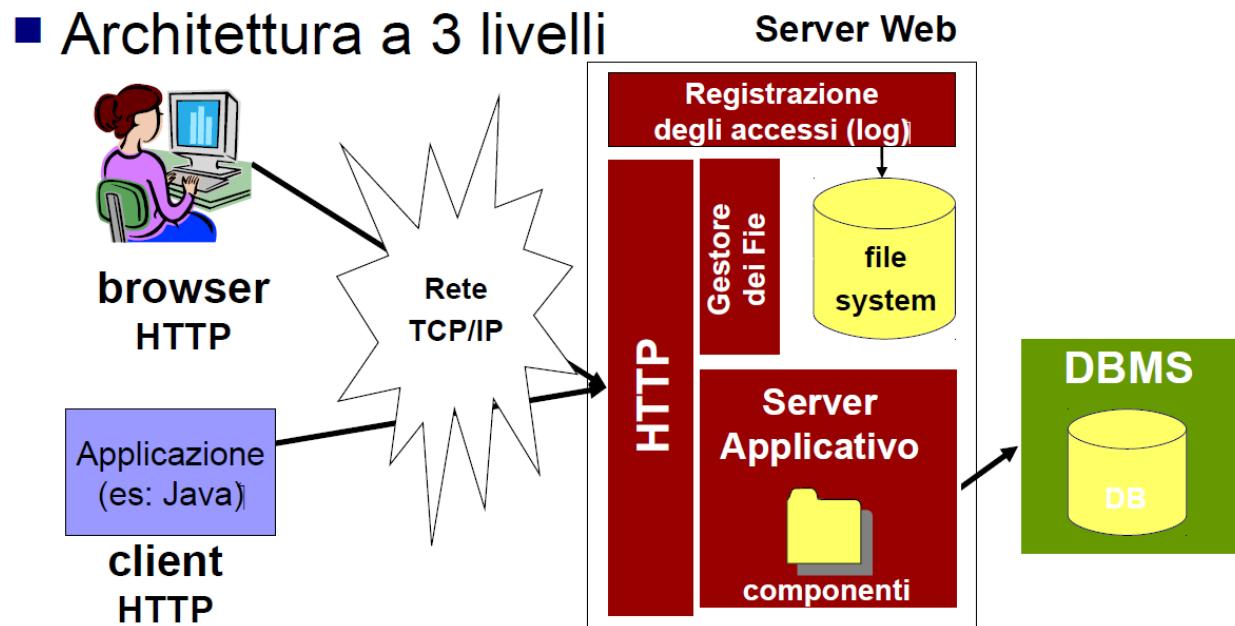
- Contiene vari moduli
- **Server HTTP**
 - implementa il protocollo HTTP
 - include vari altri servizi; es: caching, logging
- **Gestore del file system**
 - contenuti statici salvati come file
- **Server applicativo**
 - gestore di applicazioni e componenti

■ Architettura a 3 livelli



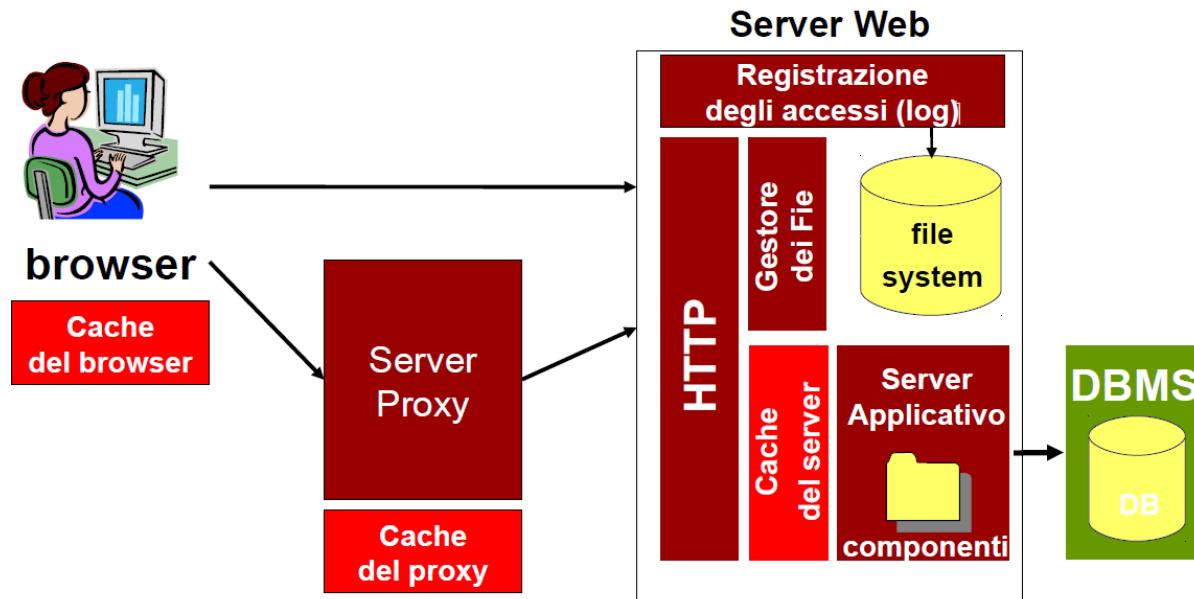
Registrazioni (Logging)

- Servizio di registrazione
 - tutto quello che avviene sul server viene registrato in opportuni file di registro (log)
- File principali
 - richieste: IP del richiedente, URI richiesto, data, esito
 - errori
 - provenienza (referer)



Caching

- **Caching**
 - Memorizzazione temporanea delle informazioni
 - Serve a migliorare le prestazioni
 - Più in generale
 - tecnica essenziale per l'accesso ai dati distribuiti
 - nell'architettura di riferimento esistono meccanismi di caching a più livelli
- Caching a vari livelli delle risposte



Uniform Resource Identifiers (URI)

- Sistema di indirizzamento su Web
 - stringhe di caratteri ASCII che identificano le risorse disponibili su Web
- Standard IETF (RFC 2396) : Internet Engineering Task Force
- **Uniform/Universal Resource Identifiers (URI)**
 - Uniform Resource Names (URN)
 - la risorsa può non essere fisicamente accessibile (es: namespace)
 - la stringa non descrive il metodo d'accesso
 - Uniform/Universal Resource Locators (URL)
 - la risorsa è fisicamente accessibile
 - la stringa descrive il metodo (primario) per accedere alla risorsa

Forma generale

<protocollo>:<parte-dipendente-dal-protocollo>

Principali protocolli

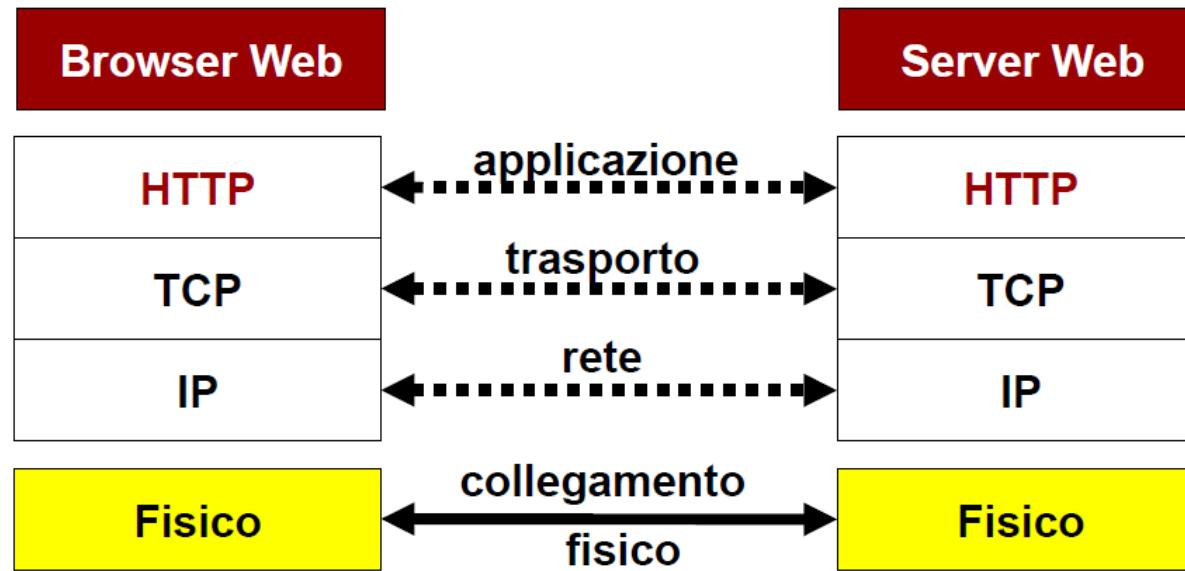
http, ftp, mailto, file

Esempi:

http://www.repubblica.it/sport/
ftp://ftp.unipr.it/pub
mailto:vincenzo.bonnici@unipr.it
file:///d:/sites/index.html
gopher://spinaltap.micro.umn.edu/00/
news:comp.infosystems.www.servers.unix
telnet://melvyl.ucop.edu/

HTTP 1.0

- Standard IETF (RFC 1945) ora superato da HTTP 1.1
- Protocollo di applicazione



- **Transazione HTTP**

- scambio di messaggi tra server e client
- il client apre una connessione con il server
- il client invia la richiesta sulla connessione
- il server invia la risposta sulla connessione
- la connessione viene chiusa

- **Caratteristiche del protocollo**

- una nuova connessione per ogni transazione
 - Migliorato in HTTP 1.1
- **privo di stato** – nella transazione successiva non resta traccia di quanto avvenuto nelle transazioni precedenti

- Attenzione

- la mancanza di stato influenza la scrittura delle applicazioni
 - Cookies, sessioni

HTTP 1.0 : Transazioni

Come nasce normalmente la richiesta

- l'utente seleziona un URI nel browser
- es: http://www.sci.unich.it/~fioravan
- l'URI può essere specificato esplicitamente
es: nella barra degli indirizzi del browser
- oppure può provenire da un collegamento ipertestuale selezionato dall'utente

I Operazione

- risoluzione del nome: il client utilizza il servizio
- DNS per risolvere il nome in num. IP
- es: www.sci.unich.it >> 192.167.92.35

II Operazione

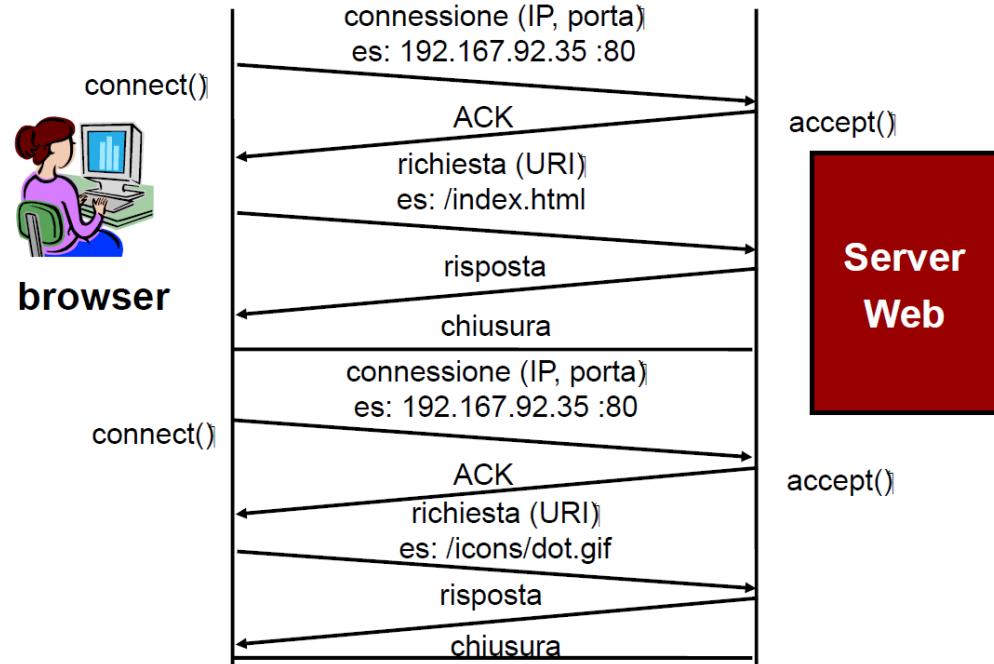
- viene richiesta una connessione al numero IP
- e alla porta specificata
- es: 192.167.92.35:80

III Operazione

- ottenuta la connessione, il browser effettua
- una richiesta HTTP al server specificando il percorso e il nome della risorsa
- es: GET /fioravan/index.html HTTP/1.0

IV Operazione

- il server gestisce la richiesta e fornisce la risposta



Nota

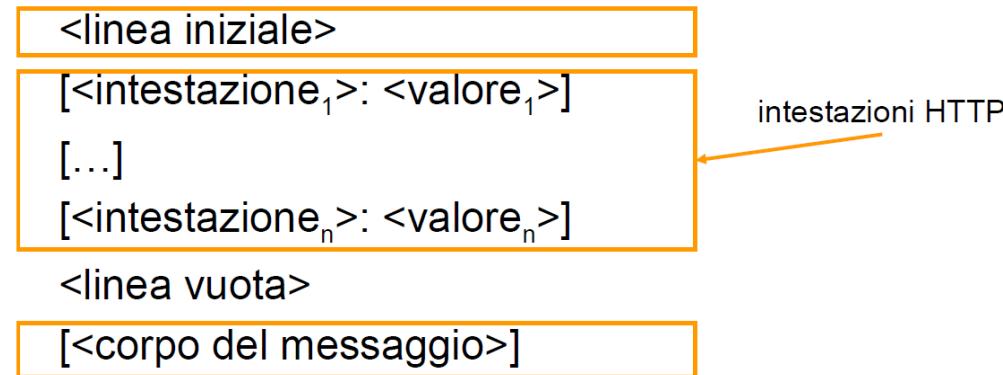
- le richieste HTTP sono raramente isolate

Esempio

- pagine HTML che contengono immagini
- il codice HTML della pagina e le immagini sono risorse distinte, con URI distinti
- viene richiesto il codice HTML
- successivamente vengono richieste le immagini necessarie alla visualizzazione completa

Struttura generale dei messaggi

vale sia per la richiesta HTTP che per la risposta HTTP



Formato dei Messaggi

- **Linea iniziale**
 - nella **richiesta**: contiene operazione e risorsa
 - nella **risposta**: contiene l'esito dell'operazione richiesta
-
- **Corpo**
 - nella richiesta è vuota o contiene la query
 - nella risposta contiene la risorsa
-
- **Intestazioni**
 - ne esistono varie

Linea iniziale della richiesta

- <metodo> <URI> HTTP/1.0

Metodo

- **GET**: metodo ordinario per effettuare richieste specificando l'URI della risorsa
 - **POST**: metodo per effettuare richieste specificando l'URI ed una serie di dati e parametri nel corpo della richiesta
 - **HEAD**: variante di GET a scopo di controllo
-

Metodo **GET**

- metodo molto comune
- viene specificato l'URI della risorsa
- eventuali parametri sono nell'URI come coppie parametro=valore
- il corpo della richiesta è vuoto

GET /index.html HTTP/1.0

GET /didattica/index.html HTTP/1.0

GET /bollo.cgi?targa=AB123DE HTTP/1.0

GET /bollo.cgi?CF=19&alimentazione=benzina HTTP/1.0

Metodo **POST**

- spesso utilizzato per inviare dati tramite moduli HTML (form)
- viene specificato l'URI della risorsa senza parametri
- parametri contenuti nel corpo del messaggio

POST /bollo.cgi HTTP/1.0 (in questo caso i parametri sono nel corpo)

Metodo **HEAD**

- variante di GET utilizzata principalmente a scopo di controllo (es: validità) e debugging
- la richiesta è del tutto simile ad una GET
- in risposta il server fornisce solo le intestazioni (ma non il corpo)

HEAD /index.html HTTP/1.0

HEAD /bollo.cgi?targa=AB123DE HTTP/1.0

Intestazioni delle **richieste** HTTP

- User-Agent – es: User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; Q312461)
 - Quale tipo e versione di browser si usa
- If-Modified-Since – es:
 - If-Modified-Since: Thu, 01 Mar 2013 08:00:00 GMT
 - Solo risorse modificate dopo una certa data
- Authorization – es:
 - Authorization: Basic ZGRpbjpvcGVuIHNl==
 - Referer – es. Referer: <http://www.unipr.it/index.html>

Intestazioni delle **risposte** HTTP

- Content-Type – es: Content-Type: text/html
- Content-Length – es: Content-Length: 650
- Last-Modified –
 - es: Last-Modified Thu, 01 Apr 2002 16:00:00 GMT
- Pragma – es: Pragma: no-cache
- Server – es: Server: Apache 1.3.20
- Location –
 - es: Location: http://www.unich.it/newindex.html
- WWW-Authenticate –
 - es: WWW-Authenticate: Basic realm="Area Privata"

GET

■ Richiesta

GET /news/index.html HTTP/1.0

User-Agent: Mozilla/4.0
(compatible; MSIE 5.0;
Windows XP) Opera 6.0 [en]

Referer: http://www.unich.it

<linea vuota>

■ Risposta

HTTP/1.0 200 OK

Date: Thu, 01 Apr 2002 16:00:00 GMT
Content-Type: text/html
Content-Length: 1534

```
<html>
<head>
...
</head>
<body>
...
</body>
</html>
```

corpo della
risposta:
contenuto del
file index.html

POST

■ Richiesta

POST /bollo.asp HTTP/1.0

User-Agent: Mozilla/4.0
(compatible; MSIE 5.0;
Windows XP) Opera 6.0 [en]

targa=AB123CD&utente=Mario
%20Rossi

si suppone che l'utente abbia
riempito e sottomesso una maschera
basata sul metodo POST

■ Risposta

HTTP/1.0 200 OK

Date: Thu, 01 Apr 2002 16:00:00
GMT

Content-Type: text/html
Content-Length: 2384
Pragma: no-cache

```
<html>
...
targa: AB123CD
EUR 110,00 ...
</html>
```

corpo della
risposta:
codice HTML
generato
dinamicam.

Attenzione alle differenze

- nel primo caso stiamo richiedendo il **contenuto** di un file (`index.html`)
- nel secondo caso stiamo chiedendo **l'esecuzione** di un'applicazione, passando dei parametri
 - l'applicazione genera il codice HTML corrispondente al messaggio di risposta
- Ma, tutte e due i tipo di richiesta possono essere fatti sia in GET che in POST

Novità principali

- **connessioni persistenti**
 - più di una transazione si può svolgere lungo la stessa connessione TCP
 - il server può chiudere la connessione unilateralmente dopo un certo “timeout”
- **host virtuali**
 - Ad uno stesso IP possono corrispondere nomi diversi e server diversi
- **autenticazione crittografata (“digest”)**
 - Le password non vengono trasmesse
 - Il server invia al browser una stringa “nonce”, quindi il browser richiede nome utente e password all’utente
 - non è sicuro al 100%: è possibile intercettare la richiesta e riprodurla per accedere alle risorse protette
 - HTTP over SSL (Secure Socket Layer) (RFC 2818)
 - soluzione considerata più sicura
 - protocollo di trasporto; tutti i messaggi sono crittografati; crittografia a chiave pubblica (certificato); trasparente per lo sviluppatore

Altre novità

- **nuovi metodi di accesso: PUT, DELETE, ...**
- miglioramento dei meccanismi di caching

- Righe testuali free-format che specificano caratteristiche
 - generali della trasmissione (header generali)
 - dell'entità trasmessa (header di entità)
 - della richiesta effettuata (header di richiesta)
 - della risposta generata (header di risposta)

- **Header generali**
 - Si applicano solo al messaggio trasmesso e si applicano sia ad una richiesta che ad una risposta, ma non necessariamente alla risorsa trasmessa
 - Date: per data ed ora della trasmissione
 - RFC 1123 per il formato della data (possibili anche altri formate)
 - Pragma: no-cache per risposta direttamente dall'origin server (no copia in cache di qualche proxy)
- **Header di entità**
 - Forniscono informazioni sul corpo del messaggio, o, se non vi è corpo, sulla risorsa specificata
 - Content-Type: il tipo MIME dell'entità acclusa
 - Header obbligatorio in ogni messaggio che abbia un corpo
 - Content-Length: la lunghezza in byte del corpo

- **Multipurpose Internet Mail Extensions**
 - Usati originariamente per caratterizzare la parte NON ASCII delle mail
 - Definizione originale RFC2046
- Formato semplice: **type / media type name [+suffix]**
- Esempi - http://en.wikipedia.org/wiki/Internet_media_type
 - text/html
 - text/plain
 - text/css
 - application/javascript
 - application/json
 - application/zip
 - application/xhtml+xml
 - application/vnd.ms-excel
 - audio/mpeg
 - video/mpeg

- From
 - From: pierpaolo.lore3@uniroma2.it
- User-Agent
 - User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36
- If-Modified-Since
 - If-Modified-Since: Wed, 21 Oct 2015 07:28:00 GMT
- Referer
 - Referer: <http://www.uniroma2.it/>
- Authorization
 - Authorization: Basic QWxhZGRpbjpPcGVuU2VzYW1l

Header di risposta

- Server
 - Server: Apache 1.3.20
- Location
 - Location: http://web.uniroma2.it/
- WWW-Authenticate
 - WWW-Authenticate: Basic realm=“Area Privata”

Response status code

- Status-codes 1xx - Informational

- Riservato per usi futuri

- Status-codes 2xx - Success

- L'azione è stata ricevuta con successo, capita e accettata
 - 200 OK
 - 201 POST command successful
 - 202 Request accepted
 - 203 GET or HEAD request fulfilled
 - 204 No content

- Status-codes 3xx - Redirection

- Per completare la richiesta è necessaria un'altra azione
 - 300 Resource found at multiple locations
 - 301 Resource moved permanently
 - 302 Resource moved temporarily
 - 304 Resource has not modified (since date)

- Status-codes 4xx - Client error

- La richiesta contiene una sintassi errata o non può essere servita
 - 400 Bad request from client
 - 401 Unauthorized request
 - 402 Payment required for request
 - 403 Resource access forbidden
 - **404 Resource not found**
 - 405 Method not allowed for resource
 - 406 Resource type not acceptable

- Status-codes 5xx - Server error

- Il server non riesce a servire una richiesta apparentemente valida
 - 500 Internal server error
 - 501 Method not implemented
 - 502 Bad gateway or server overload
 - 503 Service unavailable / gateway timeout
 - 504 Secondary gateway / server timeout

- Content-Type
 - Content-Type: text/html
- Content-Length
 - Content-Length: 650
 - Tipicamente omesso in risposte contenenti risorse generate dinamicamente
- Content-Encoding
 - Content-Encoding: gzip
 - Tipo di compressione applicato alla risorsa
- Allow
 - Allow: GET
 - Metodi supportati dalla risorsa specificata dall'URI
- Last-Modified
 - Last-Modified: Thu, 01 Apr 2002 18:16:45 GMT
- Expires
 - Expires: Fri, 02 Apr 2002 21:00:00 GMT

PHP request header

- Memorizza1 nella variabile \$_SERVER
 - Host -> HTTP_HOST
 - Connec1on -> HTTP_CONNECTION
 - ...
- in PHP >5.6
 - getallheaders()

PHP response header

- L'header è gestito da PHP ed Apache trattando l'output dello script come HTML
- Con la funzione **header()** possiamo gestire l'header HTTP
 - per esempio ges1re il Content-Type o response code

```
<?php header("Content-Type: text/plain"); ?>  
Date: today  
From: fred  
To: barney  
Subject: hands off!  
My lunchbox is mine and mine alone.  
Get your own, you filthy scroung
```

```
<?php header("Content-Type: text/plain"); ?>
```



```
<?php  
header($_SERVER["SERVER_PROTOCOL"]." 404 Not Found");  
echo "Pagina non trovata";
```

- Redirection

```
<?php
header('HTTP/1.1 301 Moved Permanently');
header("Location: http://ppl.eln.uniroma2.it/pw/");
exit();
```

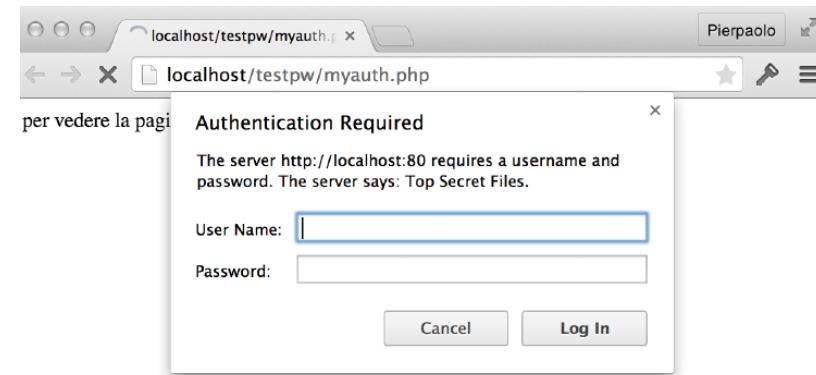
- Authentication

```
header('WWW-Authenticate: Basic realm="Top Secret Files"');
header('HTTP/1.0 401 Unauthorized');
```

```
<?php
$authOK = false;
$user = $_SERVER['PHP_AUTH_USER'];
$password = $_SERVER['PHP_AUTH_PW'];

//Controllo autenticazione
if (isset($user) && isset($password)
    && $user === "loreti" && $password === "ciao") {
    $authOK = true;
}

if (!$authOK) {
    header('WWW-Authenticate: Basic realm="Top Secret Files"');
    header('HTTP/1.0 401 Unauthorized');
    // Ciò che stampiamo qui è visto da chi clicca cancel
    exit;
}
?>
<h1>Pagina protetta</h1>
```



Expire

```
// Set the expire header
header("Expires: Fri, 18 Jan 2006 05:30:00 GMT");

// Set the expire a 3 ore
$now = time();
$then = gmstrftime("%a, %d %b %Y %H:%M:%S GMT", $now + 60 * 60 * 3);
header("Expires: {$then}");

// Indicare che il documento non perde di validità mai
$now = time();
$then = gmstrftime("%a, %d %b %Y %H:%M:%S GMT", $now + 365 * 86440);
header("Expires: {$then}");

// Inibire il caching di un documento
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
header("Cache-Control: no-store, no-cache, must-revalidate");
header("Cache-Control: post-check=0, pre-check=0", false);
header("Pragma: no-cache");
```

- Tutti gli Header HTTP hanno la lettere maiuscola
 - Location, not location
 - Content-Type, not content-type, nor CONTENT-TYPE
- Dopo i due punti c'è uno spazio
 - good: header("Content-Type: text/plain");
 - wrong: header("Content-Type:text/plain");
- Usate URI assolute e complete di schema dominio, nei Location header
 - good: header("Location: http://www.example.com/something.php?a=1");
- Le URI relative non sono permesse
 - wrong: Location: /something.php?a=1
 - wrong: Location: ?a=1

Errori comuni

- Iniziare il file con qualcosa diverso da <?php
 - anche una riga vuota o uno spazi

The screenshot shows a PHP code editor with the following code:

```
<?php  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11
```

A warning message is displayed in the status bar:

(!) Warning: Cannot modify header information - headers already sent by (output started at /Users/loreti/htdocs/testpw/testcommonerr.php:1) in /Users/loreti/htdocs/testpw/testcommonerr.php on line 1

Call Stack:

#	Time	Memory	Function	Location
1	0.0002	322892	{main}()	./testcommonerr.php:0
2	0.0002	323064	header()	./testcommonerr.php:1

Includere un file che termina con una riga vuota o altro causa l'invio dell'header

- chiudo il file con ?>
- non chiudo il php

The screenshot shows a PHP code editor with the following code:

```
<?php  
2 include 'libreria.php.inc';  
3 <?php
```

A warning message is displayed in the status bar:

(!) Warning: Cannot modify header information - headers already sent by (output started at /Users/loreti/htdocs/testpw/libreria.php.inc:10) in /Users/loreti/htdocs/testpw/testec2.php on line 4

Call Stack:

#	Time	Memory	Function	Location
1	0.0058	322912	{main}()	./testec2.php:0
2	0.0493	323764	header()	./testec2.php:4

Download di file

```
<?php
$file = 'file.doc';

if (file_exists($file)) {
    header('Content-Description: File Transfer');
    header('Content-Type: application/octet-stream');
    header('Content-Disposition: attachment; filename=' . basename($file));
    header('Expires: 0');
    header('Cache-Control: must-revalidate');
    header('Pragma: public');
    header('Content-Length: ' . filesize($file));
    readfile($file);
    exit;
}
```

- **HTTP è stateless**

- ogni richiesta genera un processo serve una sola richiesta e poi termini
- ogni richiesta http è separata dalle altre

Idea: Il server manda "qualcosa" al client che lo re-invia al server nelle richieste successive per collegare fra loro le richieste

- Tre tecniche

- **hidden form**

- le variabili si inseriscono in campi hidden di un form
 - uso molto limitato

- **url parameter**

- le variabili si concatenano alle url dei link
 - h:p://www.example.com/catalog.php?userid=123

- **cookie**

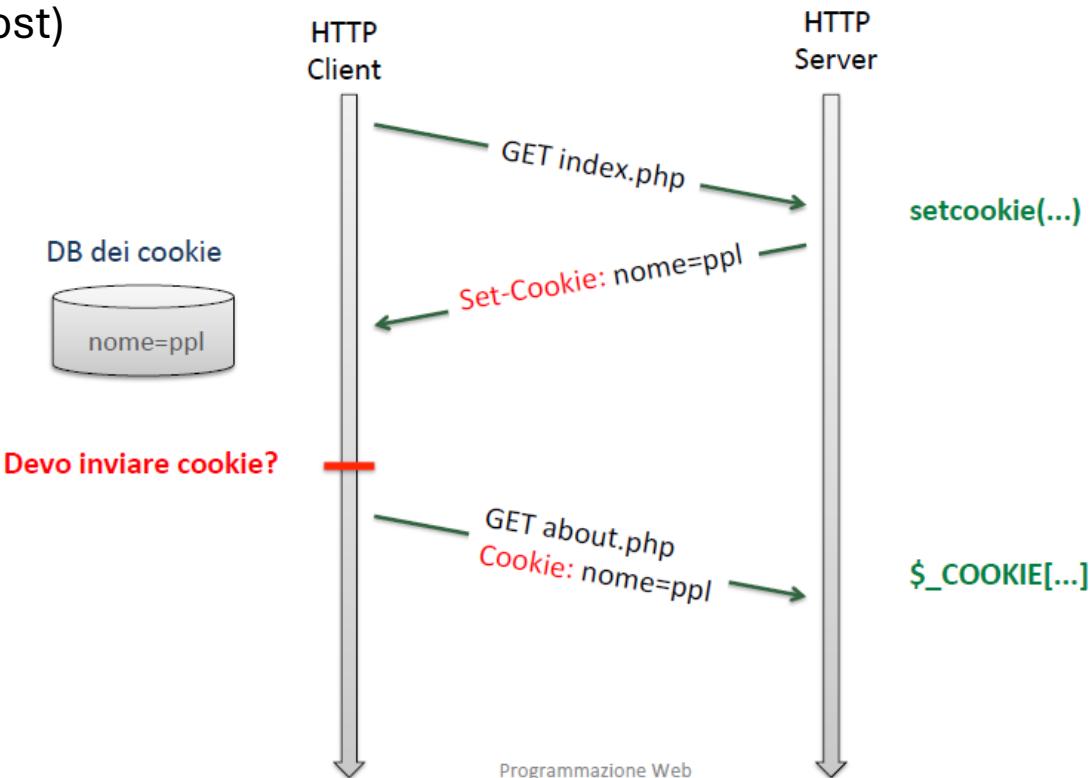
- variabili passate nell'header di req e resp
 - il funzionamento dipende dalla configurazione del browser

- **PHP fornisce le sessioni**

- default sono basate su **cookie**, ma possono lavorare anche con le url

Cookie

- Il **cookie** è una stringa in cui vengono salvati i dati del server memorizzati dal client
- Funzionamento
 - Il server setta il cookie nella **risposta** HTTP
 - **Set-Cookie:** userid=123
 - I browser lo memorizza
 - Quando fa una nuova **richiesta** lo invia
 - **Cookie:** userid=123
 - Nella richiesta (get o post)



- **setcookie**

- imposta il cookie nell'header HTTP della risposta

- setcookie(name [, value [, expire [, path [, domain [, secure]]]]);**

- va chiamata prima che l'header venga generato

- **\$_COOKIE**

- Permette di accedere ai cookie
 - si usa il nome del cookie come indice
 - **\$_COOKIE['name']**
 - il contenuto è il valore

```
<?php  
$pageAccesses = $_COOKIE['accessi'];  
setcookie('accessi', ++$pageAccesses);  
  
echo "Hai fatto $pageAccesses accessi clicca per aumentarli<br>";  
  
$actual_link = 'http://'. $_SERVER['HTTP_HOST']. $_SERVER['PHP_SELF'];  
echo "<a href=\"$actual_link\">Reload</a>";
```

▼ Request Headers view parsed
GET /testpw/testcoockie.php HTTP/1.1
Host: localhost
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/42.0.2311.135 Safari/537.36
Accept-Encoding: gzip, deflate, sdch
Accept-Language: it,en-US;q=0.8,en;q=0.6
Cookie: accessi=18

▼ Response Headers view parsed
HTTP/1.1 200 OK
Date: Thu, 14 May 2015 20:13:12 GMT
Server: Apache/2.2.14 (Unix) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.8l PHP/5.3.1 mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.1
Set-Cookie: accessi=19
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html

localhost/testpw/testcoockie.php

Hai fatto 19 accessi clicca per aumentarli
[Reload](#)

Parametri di setcookie

- expires
 - Specifica la durata del cookie (la data di scadenza).
 - Se indicata ed è successiva alla data settata sul sistema dell'utente, il cookie verrà memorizzato sul disco dell'utente
 - Se non è indicata, il cookie associato al documento dura solo fino alla chiusura della sessione del browser (risiede in RAM)
- path
 - Specifica la directory sul server a cui è associato il cookie • es. path=/corsi/
 - Il cookie sarà accessibile da tutte le URL residente nel sottoalbero radicato in /corsi/
 - Il valore di default è il path della pagina che sta impostando il cookie
- domain
 - Indica il dominio in cui il cookie è visibile
 - Se “domain=uniroma2.it” e “path=/” il cookie è visibile anche in www.eln.uniroma2.it, www.ing.uniroma2.it, webmail.uniroma2.it, ...
 - default: host name del server
- secure
 - indica che il cookie deve essere inviato solo quando browser e server sono connessi tramite HTTPS od un altro protocollo sicuro

Cosa invia il browser?

- Gli attributi expires, path, domain e secure sono utilizzati dal client per determinare se inviare o meno il cookie indietro ad un server
- Il browser invierà al server solo nome=valore per ogni cookie non scaduto che “corrisponde” al sito e path visitato

Limiti

Dipendono dal browser, comunque la specifica (RFC 2109, HTTP State Management Mechanism) impone le seguenti limitazioni:

- Grandezza cookie: 4KB
- Numero di cookie per sito: 20
- Numero totale di cookie: 300
 - Se il browser ha già memorizzato il numero massimo di cookie, ne cancella uno non ancora scaduto per memorizzarne uno nuovo

Array e valori multipli nei coockie

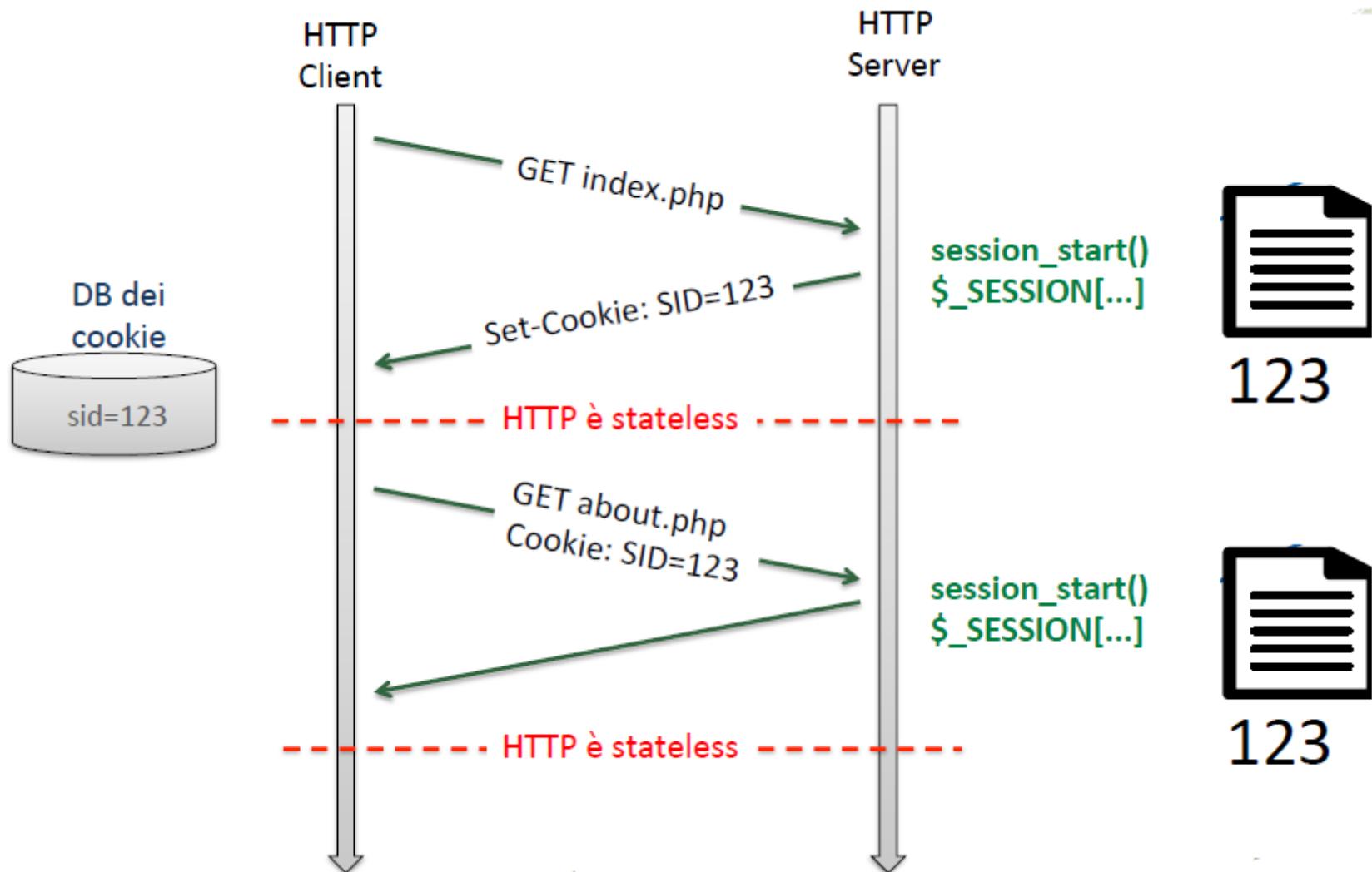
- Si può registrare un array in un cookie usando la notazione degli array al posto del nome del cookie
 - Questo equivale alla spedizione di tanti cookie quanti sono gli elementi dell'array
 - Quando il cookie è ricevuto, tutti i suoi valori sono ordinati in un singolo array che ha per nome il nome del cookie
-
- Il cookie può conservare solo un'informazione relativa all'utente che visita la pagina
 - nome/valore
 - Per inserire più coppie nome/valore bisogna devo formattare opportunamente il valore
 - Esempio di formattazione:
 - Le coppie di nome/valore sono separate da &; mentre, nome e valore sono separati da =
 - La funzione che leggerà il cookie deve elaborare la stringa tenendo conto della formattazione

```
setcookie("test","nome=pierpaolo&id=1234");
```

Cancellare un cookie

- Si imposta il cookie con l'expire scaduto
 - setcookie ("nome", "", 0);
 - setcookie ("nome", "", time()-1);
 - potrebbe fallire dipende dall'ora del client

- PHP consente di salvare un certo numero di variabili e “propagarle” da una pagina all’altra
- I valori delle variabili sono memorizzate sul server e non sui client
- Il client memorizza l'id di sessione
- Le sessioni trasformano di fatto l'HTTP da STATELESS a STATEFULL, consentendo applicativi come “carrelli della spesa”, forum, quiz on line, ecc...
- PHP crea un identificativo di sessione e lo memorizza con un cookie
 - può essere propagato anche attraverso i parametri delle URL
 - l'identificativo è un numero "unico"
- random con bassa probabilità di collisione
- I valori delle variabili di sessione sono memorizzate in un file presente sul server
 - un file per ciascuna sessione con nome id della sessione



Gestire le sessioni

- **session_start()**

- Avvia la sessione
- se è stato passato un **ID** usa quello altrimenti lo crea
- \$a=**session_id()**; per recuperare l'ID di sessione
- Va eseguito all'inizio dello script o prima di utilizzare le variabili di sessione

- **\$_SESSION['key']**

- Registra la variabile key come variabile di sessione
- Mantiene il suo valore fra chiamate http se la sessione è gestita

```
<?php  
  
session_start();  
$_SESSION['hits'] = $_SESSION['hits'] + 1;  
echo "This page has been viewed {$_SESSION['hits']} times.";
```

Response Headers [view source](#)

Cache-Control: no-store, no-cache, must-revalidate, post-check=0
Connection: Keep-Alive
Content-Type: text/html
Date: Thu, 14 May 2015 21:26:36 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Keep-Alive: timeout=5, max=100
Pragma: no-cache
Server: Apache/2.2.14 (Unix) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.81
Set-Cookie: PHPSESSID=b6e2d134706a075153466013d8939f7f; path=/
Transfer-Encoding: chunked
X-Powered-By: PHP/5.3.1

localhost/testpw/testsession.php
This page has been viewed 3 times.

Request Headers [view source](#)

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/*,
Accept-Encoding: gzip, deflate, sdch
Accept-Language: it,en-US;q=0.8,en;q=0.6
Cache-Control: no-cache
Connection: keep-alive
Cookie: PHPSESSID=b6e2d134706a075153466013d8939f7f
Host: localhost
Pragma: no-cache
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36

Sessioni con url

```
<?php

ini_set("session.use_cookies",0);
ini_set("session.use_only_cookies",0);
ini_set("session.use_trans_sid",1);

session_start();
$_SESSION['hits'] = $_SESSION['hits'] + 1;

echo "<p>Hai visitato questa pagina {$_SESSION['hits']} volte.</p>";
?>
<p>
    <a href='testsessionurl.php'>reload</a>.
</p>
```



- **session_destroy()**

- Distrugge la sessione in corso e tuye le informazioni di sessione associate (variabili di sessione, ecc)
- Non cancella il cookie settato nella cache del browser dell’utente che ha invocato lo script
- Restituisce TRUE in caso di successo e FALSE in caso di fallimento nel distruggere i dati di sessione
- Non cancella i valori delle variabili all’interno della RAM

- **session_unset();**

- Elimina i valori nella variabile \$_SESSION

- **session_set_cookie_params**

- Imposta i parametri per la sessione

```
void session_set_cookie_params ( int $lifetime [, string $path [, string $domain [, bool $secure  
= false [, bool $httponly = false ]]]] )
```

- life.me: Lifetime of the session cookie, defined in seconds.
 - path: Path on the domain where the cookie will work. Use a single slash ('/') for all paths on the domain.
 - domain: Cookie domain, for example 'www.php.net'. To make cookies visible on all subdomains then the domain must be prefixed with a dot like '.php.net'.
 - secure: If TRUE cookie will only be sent over secure connections.
 - httponly: If set to TRUE then PHP will attempt to send the httponly flag when sevng the session cookie.

```
class Person {  
  
    var $name;  
  
    function stampa() {  
        echo "Hi ".$this->name."!!\n";  
    }  
  
}  
  
$io = new Person();  
$io->name = "Pierpaolo";  
$io->stampa();  
  
$tu = new Person();  
$tu->name = "Andrea";  
$tu->stampa();
```

```
class Person {  
  
    private $name;  
  
    function __construct($name) {  
        $this->name=$name;  
    }  
  
    function stampa() {  
        echo "Hi ".$this->name."!!\n";  
    }  
  
}  
  
$io = new Person("Pierpaolo");  
$io->stampa();  
$tu = new Person("Andrea");  
$tu->stampa();  
  
// $io->name = "Paolo"; genera un errore
```

```
class Student extends Person{
    private $id;
    function __construct($id,$name) {
        parent::__construct($name);
        $this->id=$id;
    }
}

$io = new Student(1,"Roberto");
$tu = new Student(2,"Mario");

var_dump($io);
var_dump($tu);
```

```
object(Student)[1]
private 'id' => int 1
private 'name' (Person) => string 'Roberto' (length=7)

object(Student)[2]
private 'id' => int 2
private 'name' (Person) => string 'Mario' (length=5)
```

Altro

- Incapsulamento
 - private, protected
- Overriding
 - metodi virtuali
- Interfacce
 - polimorfismo dei dati
- Trait
 - Ereditarietà multipla

Eccezioni in PHP

- Gestione degli errori OO
- Un modo per cambiare il flusso normale del codice se una condizione eccezionale (errore) accade
- Uso base
 - Porzioni di codice protette
 - Codice per gestire l'eccezione

```
<?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);
```

(!) Fatal error: Uncaught exception 'Exception' with message 'Value must be 1 or below' in /Users/loreti/Dropbox/Didattica/Corso PW/esempi/PHP-Esempi/eccezioni/eccezioni.php on line 12															
(!) Exception: Value must be 1 or below in /Users/loreti/Dropbox/Didattica/Corso PW/esempi/PHP-Esempi/eccezioni/eccezioni.php on line 12															
Call Stack															
<table border="1"><thead><tr><th>#</th><th>Time</th><th>Memory</th><th>Function</th><th>Location</th></tr></thead><tbody><tr><td>1</td><td>0.0001</td><td>230144</td><td>{main}()</td><td>./eccezioni.php:0</td></tr><tr><td>2</td><td>0.0001</td><td>230752</td><td>checkNum()</td><td>./eccezioni.php:18</td></tr></tbody></table>	#	Time	Memory	Function	Location	1	0.0001	230144	{main}()	./eccezioni.php:0	2	0.0001	230752	checkNum()	./eccezioni.php:18
#	Time	Memory	Function	Location											
1	0.0001	230144	{main}()	./eccezioni.php:0											
2	0.0001	230752	checkNum()	./eccezioni.php:18											

Eccezioni in PHP

```
function checkNum($number){  
    if($number>1) {  
        throw new Exception("Value must be 1 or below");  
    }  
    return true;  
}  
  
//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 $e){  
    echo 'Message: ' . $e->getMessage();  
}
```

Eccezioni in PHP

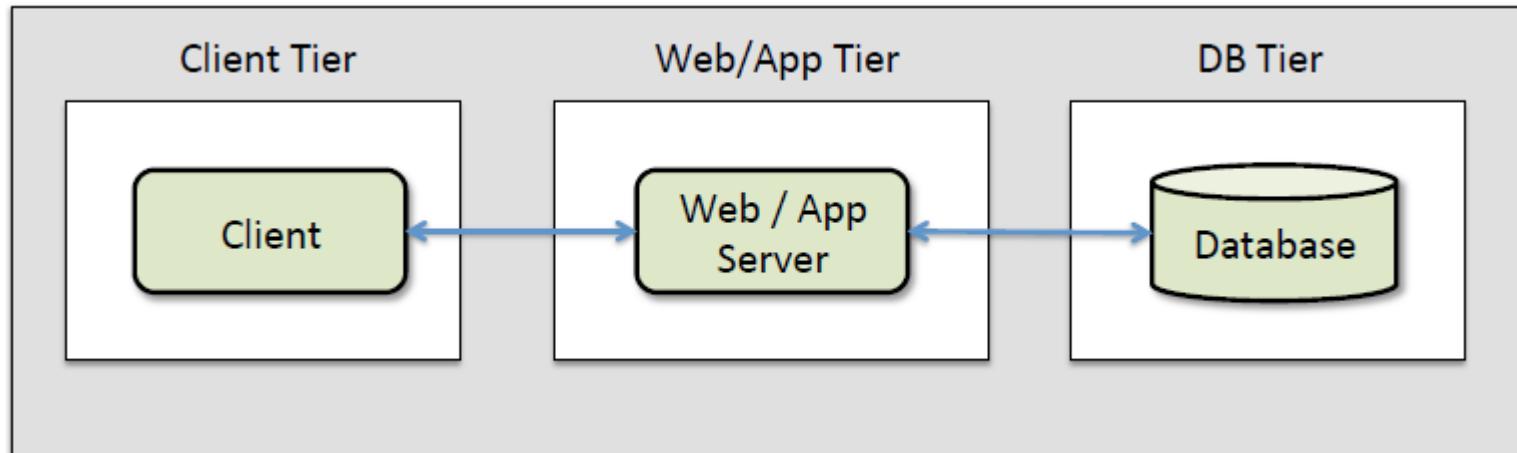
```
class MyException extends Exception {
    public function errorMessage() {
        //error message
        $errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
        .': <b>'.$this->getMessage().'</b> is not a valid E-Mail address';
        return $errorMsg;
    }
}

$email = "someone@example.com";

try {
    //check if
    if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE) {
        //throw exception if email is not valid
        throw new MyException($email);
    }
    //check for "example" in mail address
    if(strpos($email, "example") !== FALSE) {
        throw new Exception("$email is an example e-mail");
    }
} catch (MyException $e) {
    echo $e->errorMessage();
} catch(Exception $e) {
    echo $e->getMessage();
}
```

Architettura web con database

- Il server web comunica con un altro server che contiene il la banca dati
 - la comunicazione usa dei protocolli specifici



- Librerie per interagire con il db
 - **PDO** (PHP Data Objects)
 - generica per vari db
 - **MySQLi** extension (the "i" stands for improved)
 - due versioni: procedurale ed a oggetti
 - **MySQL** extension (senza i)
 - deprecata nel 2012 ma ancora molto usata

- **MySQLi vs PDO**
 - PDO gestisce 12 DB e MySQLi solo uno
 - Entrambe sono OO ma MySQLi è anche procedurale
 - Gestiscono entrambe i prepared statement
 - importante per la sicurezza
 - PDO ha i parametri "named"

Procedura di accesso al db

1. Connetersi al db e selezionare il DB
2. Costruire una query string
3. Eseguire la query.
4. Recuperare i risultati ed usarli
 - mostrarli in una pagina
5. Ripetere i passi 2-4 per le query successive
6. Disconnettesi dal DB

PDO: apertura e chiusura connessione

```
$server = "localhost";
$user = "username";
$pass = "password";
$db = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$server;dbname=$db", $user, $pass);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
    /**
     */

} catch(PDOException $e){
    echo "Error: " . $e->getMessage();
}
$conn = null; // Chiusura della connessione
```

PDO: creazione di una tabella

```
CREATE TABLE MyGuests (
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    firstname VARCHAR(30) NOT NULL,
    lastname VARCHAR(30) NOT NULL,
    email VARCHAR(50),
    reg_date TIMESTAMP
)
```

```
// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)";

// use exec() because no results are returned
$conn->exec($sql);

echo "Table MyGuests created successfully";
```

PDO: inserire i dati

```
INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')
```

```
try {
    $conn = new PDO("mysql:host=$server;dbname=$db", $user, $pass);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";
    // use exec() because no results are returned
    $conn->exec($sql);
    $last_id = $conn->lastInsertId();

    echo "New record created successfully. Last inserted ID is: " . $last_id;
}

} catch(PDOException $e){
    echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
```

PDO: inserimenti multipli

```
// begin the transaction
$conn->beginTransaction();
// our SQL statements
$conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')");
$conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com')");
$conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com')");

// commit the transaction
$conn->commit();
echo "New records created successfully";
```

PDO: leggere i dati

```
try {
    $conn = new PDO("mysql:host=$server;dbname=$db", $user, $pass);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";

    $stmt = $conn->query("SELECT id, firstname, lastname FROM MyGuests");

    $result = $stmt->fetchAll();
    var_dump($result);

} catch(PDOException $e){
    echo "Error: " . $e->getMessage();
}
$conn = null; // Chiusura della connessione
```

```
$stmt = $conn->query("SELECT id, firstname, lastname FROM MyGuests");

foreach ($stmt as $row) {
    echo $row["id"] . " - " . $row["firstname"] . " " . $row["lastname"] . "<br/>";
}
```

PDO: gestione del PDOStatement

```
$stmt->setFetchMode(PDO::FETCH_ASSOC);  
$stmt->setFetchMode(PDO::FETCH_NUM);  
$stmt->setFetchMode(PDO::FETCH_OBJ);  
  
$row = $stmt->fetch(PDO::FETCH_ASSOC);  
  
$results = $stmt->fetchAll();  
  
$num_row = $stmt->rowCount();
```

PDO: prepared statement

- Eseguire query ripetute
- Funzionamento
 - Preparare un template di query in cui alcuni valori non sono specificati indicati con un ? o un nome
 - INSERT INTO MyGuests VALUES(?, ?, ?)
 - INSERT INTO MyGuests (firstname, lastname, email) VALUES (:firstname, :lastname, :email)
 - lo script registra la query nel db che la "compila"
 - lo script inserisce i dati mancati ed esegue la query
- Confronto con query normali
 - Il tempo di esecuzione quando la query è eseguita più volte è minore
 - la capacità di rete usata è minore
 - difesa naturale da SQL injection

SQL injection

UserId:

105

```
SELECT * FROM Users WHERE UserId = 105
```

UserId:

105 or 1=1

```
SELECT * FROM Users WHERE UserId = 105 or 1=1
```

User id:

105; DROP TABLE Suppliers

```
SELECT * FROM Users WHERE UserId = 105; DROP TABLE Suppliers
```

```
$param = $conn->quote("John");
$stmt = $conn->query("SELECT id,firstname, lastname "
    . "FROM MyGuests WHERE firstname = ".$param." ");
$results = $stmt->fetchAll(PDO::FETCH_ASSOC);
var_dump($results);
```

```
$stmt = $conn->prepare("SELECT id,firstname, lastname "
    . "FROM MyGuests WHERE firstname = :firstname");
$stmt->execute(array(':firstname' => "John") );

$results = $stmt->fetchAll(PDO::FETCH_ASSOC);
var_dump($results);
```

Esempi di Update e delete

```
$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";  
  
// Prepare statement  
$stmt = $conn->prepare($sql);  
  
// execute the query  
$stmt->execute();  
  
// echo a message to say the UPDATE succeeded  
echo $stmt->rowCount() . " records UPDATED successfully";
```

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30

```
// sql to delete a record  
$sql = "DELETE FROM MyGuests WHERE id=3";  
  
// use exec() because no results are returned  
$conn->exec($sql);  
echo "Record deleted successfully";
```

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30
3	Julie	Dooley	julie@example.com	2014-10-26 10:48:23

MYSQLI OOP

```
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"] . " - Name: " .
             $row["firstname"] . " " . $row["lastname"] . "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
```

MYSQLi OOP

```
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);

// set parameters and execute
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();

$firstname = "Mary";
$lastname = "Moe";
$email = "mary@example.com";
$stmt->execute();

echo "New records created successfully";

$stmt->close();
$conn->close();
```

MYSQLi procedurale

```
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

//...

mysqli_close($conn);
```

MYSQLi procedurale

```
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

//...
mysqli_close($conn);
```

```
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com');

if (mysqli_query($conn, $sql)) {
    $last_id = mysqli_insert_id($conn);
    echo "New record created successfully. Last inserted ID is: " . $last_id;
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com');

if (mysqli_multi_query($conn, $sql)) {
    echo "New records created successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
```

MYSQLI procedurale

```
$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // output data of each row
    while($row = mysqli_fetch_array($result)) {
        echo "id: " . $row["id"] . " - Name: "
            . $row["firstname"] . " " . $row["lastname"] . "<br>";
    }
} else {
    echo "0 results";
}
```

```
mysqli_fetch_array($result, MYSQLI_NUM);

mysqli_fetch_array($result, MYSQLI_ASSOC);

// Same as mysqli_fetch_array($result, MYSQLI_NUM)
mysqli_fetch_row($result);

// Same as mysqli_fetch_array($result, MYSQLI_ASSOC)
mysqli_fetch_assoc($result);
```

```
mysqli_free_result($result);
```

MYSQLi procedurale

```
$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";  
  
if (mysqli_query($conn, $sql)) {  
    echo "Record updated successfully";  
} else {  
    echo "Error updating record: " . mysqli_error($conn);  
}
```

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30

```
$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";  
  
if (mysqli_query($conn, $sql)) {  
    echo "Record updated successfully";  
} else {  
    echo "Error updating record: " . mysqli_error($conn);  
}
```

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30
3	Julie	Dooley	julie@example.com	2014-10-26 10:48:23

Operazioni CRUD

- Acronimo che identifica le quattro operazioni di base di un sistema di storage persistente
 - Create, Read (Retrive), Update, Delete (Destroy)

Operation	HTTP	SQL
Create	PUT / POST	INSERT
Read (Retrieve)	GET	SELECT
Update (Modify)	PUT / PATCH	UPDATE
Delete (Destroy)	DELETE	DELETE

