



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Intro al Linguaggio PHP

Basi di Dati

Corso di Laurea in Informatica per il Management

Alma Mater Studiorum - Università di Bologna

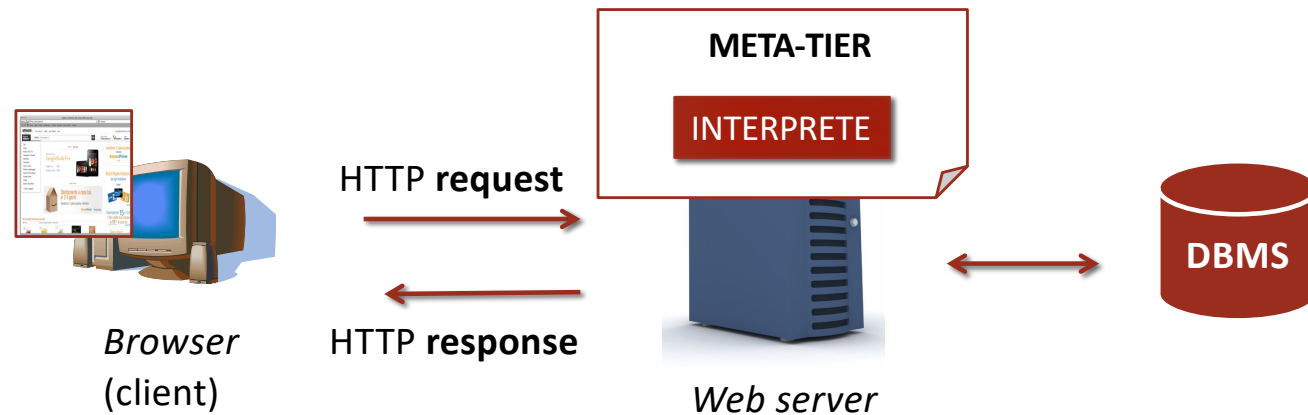
Prof. Marco Di Felice

Dipartimento di Informatica – Scienza e Ingegneria

marco.difelice3@unibo.it

Web Information System

Linguaggi di scripting server-side



TOM BUTLER, KEVIN YANK

Sviluppare applicazioni con **PHP e MySQL**



Guida per imparare
la programmazione web
lato server

APOGEO

Sviluppare applicazioni con PHP e MySQL Apogeo Editore (2018)

BASI DI DATI

PROF. MARCO DI FELICE – CORSO DI LAUREA IN INFORMATICA PER IL MANAGEMENT

Implementazione di WIS con PHP

AMP stack

- **APACHE** : Web server
- **MYSQL** : DBMS
- **PHP** : Interprete linguaggio server-side



<https://www.mamp.info/en/>

Implementazione di WIS con PHP

PHP è un **linguaggio di scripting server-side**.

Gli script PHP sono eseguiti direttamente dall' **interprete** contenuto nel web-server (Apache).

- **Sicurezza** → Gli script sono trasparenti al client.
- **Compatibilità** → Compatibilità tra client differenti.
- **Accesso a risorse** → Gli script possono accedere alle risorse contenute sul server.

Implementazione di WIS con PHP

Uno script PHP si presenta come un **insieme di istruzioni** racchiuse tra tag `<?php><?>`.

```
<html>
<head><title>Primo Script PHP</title></head>
<body>
<p>
<b> Data corrente: </b>
<?php echo date("m.d.y"); ?>
</p>
</body>
</html>
```

Implementazione di WIS con PHP

RISORSA RICHIESTA



<http://www.cs.unibo.it/data.php>

```
<html>
<head>
</head>
<body>
<p>
<b> Data corrente: </b>
<?php
echo date("m.d.y"); ?>
</p>
</body>
</html>
```

SERVER-SIDE

```
<html>
<head>
</head>
<body>
<p>
<b> Data corrente:
11.21.12</p>
</body>
</html>
```

CLIENT-SIDE

BASI DI DATI

PROF. MARCO DI FELICE – CORSO DI LAUREA IN INFORMATICA PER IL MANAGEMENT

Implementazione di WIS con PHP

PHP è un **linguaggio di programmazione** ad alto-livello:

- **Costrutti**: variabili, costrutti di selezione, iterazione, funzioni, etc
- Estensione **object-oriented**
- **Librerie** (integrazione DBMS, web-service, stringhe, etc)
- Gestione **componenti per il Web** (cookie, sessioni, etc)

Implementazione di WIS con PHP

Il comando `echo` consente di stampare una stringa (nell'HTML)

```
<html>
<head></head>
<body>
<?php
    $testVariable="<b>Salve</b>"
    echo $testVariable
?>
</body>
</html>
```



```
<html>
<head></head>
<body>
Salve
</body>
</html>
```

Implementazione di WIS con PHP

E' possibile inserire **commenti** in due modi:

- Mediante il carattere `//` (commento su singola riga)

```
$testVariable=5 // Dichiarazione
```

- Mediante il carattere: `/*` (commenti su piu righe) `*/`

```
/* Dichiarazione di una variabile  
   Commento su più righe */
```

```
$testVariable=5
```

Implementazione di WIS con PHP

PHP è un **linguaggio tipato a run-time**.

- Non è necessario dichiarare una variabile prima del suo utilizzo
(come invece avviene in Java)
- Una singola variabile può contenere un qualsiasi tipo di dato.
- I nomi delle variabili sono preceduti da \$.

```
$testVariable=5
```

```
$testVariable='Tre'
```

Implementazione di WIS con PHP

E' possibile costruire **array** eterogenei

```
$myarray=array("hello",2,3.5,"3");  
echo $myarray[0];  
echo $myarray[1];
```

E' possibile **aggiungere/modificare** elementi degli array:

```
$myarray[0]="world";  
$myarray[4]="4";  
$myarray[] = "newelem"           // Aggiunta in coda!
```

Implementazione di WIS con PHP

Gli **indici degli array** non devono essere necessariamente interi, ma possono anche essere **stringhe** (**array associativi**).

```
$birthdays["Sara"]="10/02/1982";  
$birthdays["Michele"]="12/02/1982";
```

Creazione di array associativi:

```
$birthdays=array("Sara" => "10/02/1982",  
                 "Michele" => "12/02/1982");
```

Implementazione di WIS con PHP

STRUTTURE DI CONTROLLO: **if-else**

Il comando `if-else` consente di eseguire un certo insieme di istruzioni se è soddisfatta una certa condizione (**ramo if**) o un altro insieme (**ramo else**) se la condizione non è soddisfatta.

```
if (condizione) {  
    ... Istruzioni Ramo if  
} else {  
    ... Istruzioni Ramo else  
}
```

Implementazione di WIS con PHP

STRUTTURE DI CONTROLLO: **if-else**

Il comando `if-else` consente di eseguire un certo insieme di istruzioni se e' soddisfatta una certa condizione (**ramo if**) o un altro insieme (**ramo else**) se la condizione non e' soddisfatta.

```
if ($name=="Marco") {  
    echo("Benvenuto, Marco");  
} else {  
    echo("Utente sconosciuto");  
}
```

Implementazione di WIS con PHP

STRUTTURE DI CONTROLLO: **while**

Il comando `while` consente di eseguire un certo insieme di istruzioni **nel mentre è soddisfatta una certa condizione**.

```
while (Condizione) {  
    ... Istruzione/i da ripetere  
}
```

Condizione= una qualsiasi espressione logico/matematica.

Implementazione di WIS con PHP

STRUTTURE DI CONTROLLO: **while**

Il comando `while` consente di eseguire un certo insieme di istruzioni **nel mentre è soddisfatta una certa condizione**.

```
$count=1;
while ($count<=10) {           // Stampo 10 valori
    echo($count);
    $count++;
}
echo("Fine del ciclo");
```

Implementazione di WIS con PHP

STRUTTURE DI CONTROLLO: **for**

Il comando `for` consente di eseguire un certo insieme di istruzioni **nel mentre è soddisfatta una certa condizione.**

```
for (espressione; condizione; espressione) {  
    ... Istruzione/i da ripetere  
}
```

Condizione= una qualsiasi espressione logico/matematica.

Implementazione di WIS con PHP

STRUTTURE DI CONTROLLO: **for**

Il comando `for` consente di eseguire un certo insieme di istruzioni **nel mentre è soddisfatta una certa condizione**.

```
for ($count=1; $count<=10; $count++) {  
    echo($count);           // Stampo 10 valori  
}  
echo("Fine del ciclo");
```

Implementazione di WIS con PHP

DEFINIZIONE DI FUNZIONI

Un programma PHP può includere **funzioni**, che ricevono **parametri in ingresso** e restituiscono (eventualmente) un **valore di ritorno**.

```
function somma($a, $b, $c) {  
    $result=$a + $b +$c;  
    return $result;  
}
```

```
$val=somma(1,5,6);  
echo($val)
```

Implementazione di WIS con PHP

DEFINIZIONE DI FUNZIONI

All'interno di una funzione possono essere definite **variabili locali**.
Con l'istruzione `global` si può accedere a **variabili globali** definite nello script.

```
function somma($b, $c) {  
    global $a;  
    result=$a+ $b +$c;  
    return $result;  
}  
  
$a=10;  
$val=somma(5,6);
```

Implementazione di WIS con PHP

DEFINIZIONE DI FUNZIONI

E' possibile **invocare una funzione** anche specificando un **numero di parametri inferiore a quelli previsti dall'intestazione della funzione**.

```
function f1($a, $b, $c=20) {  
    result=$a*$b+$c;  
    return $result;  
}
```

```
$a=10;  
$b=20;  
$val=f1($a,$b);
```

Implementazione di WIS con PHP

PASSAGGIO DEI PARAMETRI

Esistono **due tecniche per passare parametri in ingresso** ad uno script PHP:

- Metodo **GET**: I parametri sono inseriti nell' **URL** della risorsa.
- Metodo **POST**: I parametri sono inviati (non in forma visibile) insieme alla richiesta HTTP.

Implementazione di WIS con PHP

PASSAGGIO DEI PARAMETRI: metodo **GET**

Invio dei parametri **lato client**:

```
http://www.cs.unibo.it/script.php?nome=Michele  
http://www.cs.unibo.it/script.php?  
nome=Michele&cognome=Rossi
```

Recupero dei parametri **lato server**: (script.php)

```
$name=$_GET["nome"];  
$lastname=$_GET["cognome"];
```

\$_GET e' una variabile
di sessione di PHP contenente
i parametri nell'URL
(array associativo)

Implementazione di WIS con PHP

PASSAGGIO DEI PARAMETRI: metodo **GET**

Invio dei parametri **lato client**:

`http://www.cs.unibo.it/script.php?nome=Michele`

Utilizzo dei parametri **lato server**: (script.php)

```
<?php
$name=$_GET["nome"];
Echo("Benvenuto, ".$name."!");
?>
```

Concatenatore di stringhe

Implementazione di WIS con PHP

PASSAGGIO DEI PARAMETRI: metodo **POST**

Invio dei parametri **lato client**:

```
<form action="script.php" method="post">
  <div><label for="firstname"> Nome: </label>
  <input type="text" name="firstname" id="firstname">
</div>
<div><label for="lastname"> Cognome: </label>
  <input type="text" name="lastame" id="firstname">
</div>
<div> <input type="submit" value="Vai"></div>
</form>
```

Implementazione di WIS con PHP

PASSAGGIO DEI PARAMETRI: metodo **POST**

Recupero dei parametri **lato server**: (script.php)

```
<?php
$name=$_POST["firstname"];
$name=$_POST["lastname"];
Echo("Benvenuto, "$name.$lastname."!");
?>
```

Vantaggi rispetto al metodo GET:

(i) dimensione/numer parametri; (ii) privacy

Implementazione di WIS con PHP

PASSAGGIO DEI PARAMETRI: metodo **GET/POST**

Recupero dei parametri **lato server**: (script.php)

```
<?php
$name=$_REQUEST["firstname"];
$name=$_REQUEST["lastname"];
Echo("Benvenuto, "$name.$lastname."!");
?>
```

`$REQUEST_` astrae dalla modalità di invio parametri (GET/POST).

Implementazione di WIS con PHP

PROGRAMMAZIONE ad OGGETTI

In PHP, è possibile creare una classe con il costrutto `class`. Una classe può contenere **metodi ed attributi**.

```
class myClass {  
    public $a=10;    // Attributo  
    public function sayHello() {  
        echo("Hello! ");  
    }  
}
```

Implementazione di WIS con PHP

PROGRAMMAZIONE ad OGGETTI

Con il costrutto new, è possibile creare oggetti di una certa classe, mentre con il simbolo → è possibile **accedere a metodi/attributi di un oggetto**.

```
$myObject=new myClass();
```

```
$myObject →a = $myObject →a + 5;
```

```
$myObject →printHello();
```

Implementazione di WIS con PHP

PROGRAMMAZIONE ad OGGETTI

Se si vuole **accedere a metodi/attributi di un oggetto all'interno della definizione della classe** stessa, e' necessario usare la variabile **this** per far riferimento **all'oggetto corrente**.

```
class myClass {  
    public $a=10;    // Attributo  
    public function sayHello() {  
        echo($this→a);  
    }  
}
```

Implementazione di WIS con PHP

PROGRAMMAZIONE ad OGGETTI

Ogni classe può avere un **costruttore** (il metodo `__construct`), che può ricevere parametri in ingresso per l'inizializzazione dell'oggetto.

```
class myClass {  
    public $a=10; // Attributo  
    public function __construct($val) {  
        $this->a = $val;  
    }  
}  
myObject=new myClass(5);
```


Implementazione di WIS con PHP

PROGRAMMAZIONE ad OGGETTI

Una classe puo' disporre di **metodi ed attributi statici**, dichiarati attraverso il costrutto `static`.

```
class myClass {  
    public static $var=10; // Attributo  
    public static function hello() {  
        echo("Salve");  
    }  
}  
  
echo myClass::$var;  
myClass::hello();
```

Implementazione di WIS con PHP

PROGRAMMAZIONE ad OGGETTI

Come in altri linguaggi ad oggetti (Java, C++, etc) anche in PHP è possibile costruire **gerarchie di classi** mediante l'**ereditarietà**.

E' inoltre possibile limitare la **visibilità degli attributi/metodi** mediante gli indicatori:

- **public, private, protected**

Implementazione di WIS con PHP

PROGRAMMAZIONE ad OGGETTI

```
class A { // Definizione della classe A
    public $a=10;
    public $b=20;
    protected function get_a() {
        return $this->a;
    }
}
class B extends A { // Definizione della classe B
    protected function get_b() {
        return $this->b;
    }
}
$b=new B();
echo($b->get_a());
echo($b->get_b());
```

Implementazione di WIS con PHP

PROGRAMMAZIONE ad OGGETTI

Sfruttando l'ereditarietà tra classi, è possibile fare overriding di metodi definiti dalla classe genitore.

Altri costrutti OOP in PHP:

- Classi **astratte** (costrutto abstract)
- **Interfacce** (costrutto interface)
- **Type Hinting** (dalla versione 5 in poi ...)

Implementazione di WIS con PHP

PROGRAMMAZIONE ad OGGETTI

```
class A { // Definizione della classe A
    public $a=10;
    public $b=20;
    protected function get_a() {
        return $this->a;
    }
}
class B extends A { // Definizione della classe B
    protected function get_a() {
        return $this->a*2;
    }
}
$objectB=new B();
echo($objectB->get_a());
```

OVERRIDING

Implementazione di WIS con PHP

PROGRAMMAZIONE ad OGGETTI

```
abstract class A { // Definizione della classe A
    public $a=10;
    public $b=20;
    protected function get_a()) {
        return $this->a;
    }
    abstract protected function get_b();
}
class B extends A { // Definizione della classe B
    protected function get_b() {
        return $this->b;
    }
}
$objectB=new B();
echo($objectB->get_b());
```

CLASSI ASTRATTE

Implementazione di WIS con PHP

PROGRAMMAZIONE ad OGGETTI

```
interface A { // Definizione dell'interfaccia A
    protected function get_a();
    protected function set_a($param);
}
```

INTERFACCIA

```
class B implements A { // Definizione della classe B
    protected function get_a() {
        return $this->a;
    }
    protected function set_a($param) {
        return $this->a=$param;
    }
}
$objectB=new B();
echo($objectB->set_a(5));
```

Implementazione di WIS con PHP

CONNESSIONE CON UN DBMS

Per poter **interfacciare uno script PHP con un DBMS**, in modo da recuperare/inserire/cancellare/modificare dati da un DB, è necessario:

1. **Connettersi ad un DB**
2. Configurare la connessione
3. Costruire/Eseguire la query SQL
4. Gestire i risultati della query SQL

Implementazione di WIS con PHP

PDO (PHP Data Object): **Estensione di PHP per la connessione con un DBMS** (es. MySQL).


E' necessario creare un oggetto PDO, specificando come **parametri**:

- **Tipo** di DB (es. mysql)
- **Hostname** del server che ospita il DB
- **Nome** del DB
- **Credenziali** dell'utente (lato DBMS)

Implementazione di WIS con PHP

Esempio di connessione:

```
$pdo=new PDO("mysql:host=localhost; dbname=mydb",  
"marco", "mypassword");
```



L'istruzione ritorna **un oggetto di tipo PDO** che rappresenta la **connessione con il DB ...**

Che succede se la connessione non è disponibile?

Viene restituita **un'eccezione** da gestire...

Implementazione di WIS con PHP

Esempio di connessione con **try/catch**:

```
try {  
    $pdo=new PDO('mysql:host=localhost; dbname=mydb',  
    'marco','mypasswd');  
}  
  
catch(PDOException ex) {  
    echo("Connessione non riuscita");  
    exit();  
}
```

Implementazione di WIS con PHP

CONNESSIONE CON UN DBMS

Per poter **interfacciare uno script PHP con un DBMS**, in modo da recuperare/inserire/cancellare/modificare dati da un DB, è necessario:

1. Connettersi ad un DB
2. Configurare la connessione
3. Costruire/Eseguire la query SQL
4. Gestire i risultati della query SQL

Implementazione di WIS con PHP

Sollevare un'eccezione ogni qual volta un oggetto PDO non riesce ad eseguire un'operazione sul DB:

```
$pdo->setAttribute(PDO::ATTR_ERRMODE,  
PDO::ERRMODE_EXCEPTION);
```

Configurare la codifica dei dati:

```
$pdo->exec('SET NAMES "utf8"');
```

Implementazione di WIS con PHP

CONNESSIONE CON UN DBMS

Per poter **interfacciare uno script PHP con un DBMS**, in modo da recuperare/inserire/cancellare/modificare dati da un DB, è necessario:

1. Connettersi ad un DB
2. Configurare la connessione
3. **Costruire/Eseguire la query SQL**
4. Gestire i risultati della query SQL

Implementazione di WIS con PHP

Il comando `exec` consente di inviare una query SQL di **aggiornamento/inserimento/cancellazione** al DBMS e di eseguirla.

- **Input:** Stringa della query SQL da eseguire
- **Output:** Numero righe interessate dalla query

```
$sql="INSERT into STUDENTI VALUES ("Marco", "Rossi")";  
$result=$pdo→exec($sql);
```

Implementazione di WIS con PHP

E' necessario utilizzare try/catch in ogni query!

```
try{
    $sql='INSERT into STUDENTI VALUES
    ("Marco", "Rossi")';
    $result=$pdo→exec($sql);
}
catch(PDOException $e) {
    echo('Codice errore'.e→getMessage());
    exit();
}
echo('Numero Righe inserite:'.$result);
```


Implementazione di WIS con PHP

Il comando query consente di inviare una query SQL di **ricerca (SELECT)** al DBMS e di eseguirla.

- **Input:** Stringa della query SQL da eseguire.
- **Output:** Un oggetto PDOStatement contenente un insieme di risultati della query SQL.

```
$sql='SELECT * FROM STUDENTI';  
$result=$pdo→query($sql);
```

Implementazione di WIS con PHP

CONNESSIONE CON UN DBMS

Per poter **interfacciare uno script PHP con un DBMS**, in modo da recuperare/inserire/cancellare/modificare dati da un DB, è necessario:

1. Connettersi ad un DB
2. Configurare la connessione
3. Costruire/Eseguire la query SQL
4. **Gestire i risultati della query SQL**

Implementazione di WIS con PHP

Un oggetto di tipo `PDOStatement` è un **result set**, contenente **un elenco di tutte le righe** restituite dalla query.

- Il metodo `fetch()` consente di prelevare la riga successiva del result set.
- Se non ci sono righe, viene restituito `false` (nessuna eccezione sollevata).
- Ogni riga è un **array associativo**, con indici pari ad i nomi degli attributi della tabella risultato.

Implementazione di WIS con PHP

Esempio di query SELECT:

```
try{  
    $sql='SELECT * FROM STUDENTI';  
    $result=$pdo->query($sql);  
}  
catch(PDOException) { ...}  
  
while($row=$result->fetch()) {  
    echo( 'Nome: ' . $row[ 'name' ] );  
}
```

Oggetto PDOStatement



Implementazione di WIS con PHP

In alternativa, è possibile usare il costrutto **foreach**

```
try{
    $sql='SELECT * FROM STUDENTI';
    $result=$pdo→query($sql);
}
catch(PDOException) { ...}

foreach($result as $row) {
    echo('Nome:'. $row['name']);
}
```

Implementazione di WIS con PHP

Per ragioni di **sicurezza** (es. evitare attacchi di **SQL injection** da parte dell'utente), è meglio **evitare di inviare direttamente al DBMS una query costruita a partire da parametri** forniti dall'utente.

In alternativa:

1. Si costruisce il **template** della query.
2. Si invia il template della query al DBMS.
3. Il DBMS **prepara l'esecuzione** della query.
4. Si **inviano i parametri** utente.

Implementazione di WIS con PHP

1. Costruire il **template** della query:

```
$sql='SELECT COUNT(*) AS counter FROM Login WHERE  
(Utente=:lab1) AND (Password=:lab2)';
```

2. Inviare la query al DBMS, per **predispone l'esecuzione**:

```
$res=$pdo->prepare($sql);
```

- \$res e' un oggetto di tipo PDOStatement ..

Implementazione di WIS con PHP

3. Riempire il template della query con i **parametri** (che vanno al posto delle label definite prima):

```
$res->bindValue(":lab1",$username);
```

```
$res->bindValue(":lab2",$password);
```

4. **Eseguire** la query SQL sul DBMS

```
$res->execute();
```


Implementazione di WIS con PHP

Tramite la libreria PDO, è possibile inoltre definire operazioni SQL da eseguire all'interno di una **transazione**.

- **PDO::beginTransaction(void)**
Inizia una transazione, settando autocommit a false
- **PDO::commit()**
Effettua il commit della transazione
- **PDO::rollback()**
Richiede l'UNDO della transazione corrente

Implementazione di WIS con PHP

Il database potrebbe contenere dei **dati sensibili** (es. password di accesso), che potrebbero necessitare di **meccanismi di protezione aggiuntivi per l'accesso**.

- In PHP, e' possibile cifrare i dati con **MD5**:

```
$password=md5($_POST[ 'password' ] );
```

- Per aumentare la sicurezza, si possono aggiungere delle stringhe alla password prima di cifrarla:

```
$password=md5($_POST[ 'password' ]."jdd");
```

Implementazione di WIS con PHP

Il database potrebbe contenere dei **dati sensibili** (es. password di accesso), che potrebbero necessitare di **meccanismi di protezione aggiuntivi per l'accesso**.

- In alternativa, la stessa cifratura MD5 può essere eseguita dal DBMS (MySQL):

```
$sql='INSERT INTO Utenti(User,Password)
VALUES(“”.$username””,MD5(“”.$password””))
$result=pdo->query($sql);
```

Implementazione di WIS con PHP

A volte, l'applicazione Web ha necessità di salvare delle **informazioni di stato** sul client. Ciò può essere fatto attraverso il meccanismo dei **cookie**.

Un cookie è una **coppia <nome-valore>**, associata ad uno specifico sito web, e memorizzato sul browser del client.

1. Il cookie viene inviato dal server web al client (browser).
2. Ad ogni successiva richiesta allo stesso server, il client invia il cookie, **finchè esso non scade**.

Implementazione di WIS con PHP

1. Un browser richiede un URL cui corrisponde uno **script PHP**, al cui interno è presente la chiamata `setcookie`.
2. La pagina prodotta dallo script PHP viene spedita al client, insieme ad un header HTML contenente `<nome,valore>` del cookie.
3. Quando riceve l'header, **il browser memorizza il cookie**. Tutte le successive richieste a tale sito contengono la coppia `<nome, valore>` del cookie.

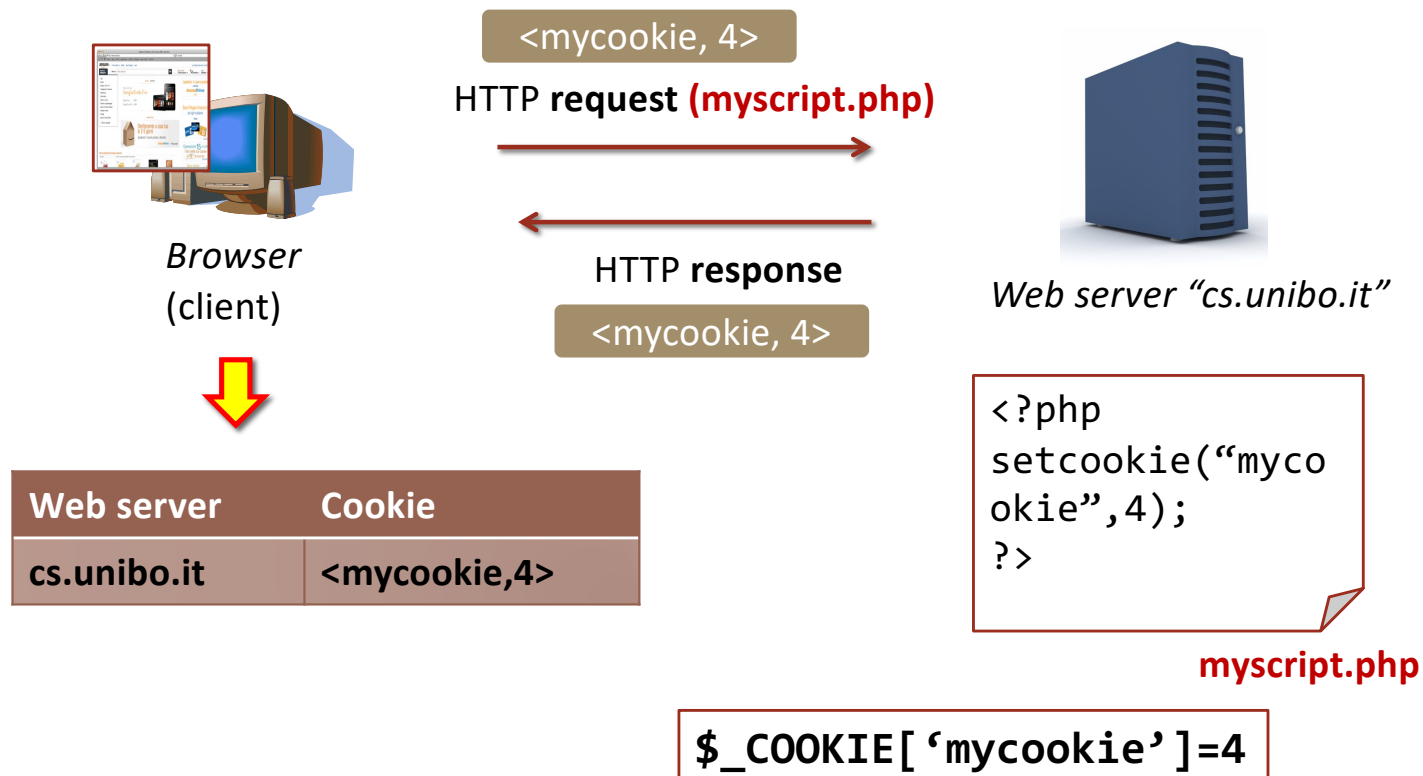
Implementazione di WIS con PHP

4. Quando riceve una richiesta con un header cookie, lo script PHP **crea una variabile** nell'array `$_COOKIE` con il **nome del cookie in questione, ed il relativo valore**.

Es. `echo $_COOKIE['mycookie_name']`

Q. Come **creare un cookie** in uno script PHP?

Implementazione di WIS con PHP



Implementazione di WIS con PHP

`setcookie(name, value, expiryTime, path, domain, secure, httpOnly)`

- Tutti i parametri sono opzionali, tranne **name**.
- **expiryTime** specifica la validità temporale del cookie (numero secondi a partire dal 1/1/1970).
- **secure** indica se il cookie deve essere inviato solo su connessioni sicure (https).

Implementazione di WIS con PHP

Q. Dove inserire la chiamata **setcookie**?

A. I **cookie** sono inviati tra gli header HTTP della pagina, quindi **setcookie** deve essere chiamata prima che venga inviato un solo byte sulla connessione!

```
<html> </head> </head>  
<body>  
>? php setcookie('mycookie'); <?>  
</body></html>
```

ERRORE!

Implementazione di WIS con PHP

PROBLEMA: I **cookie** consentono di memorizzare solo piccole quantità di dati...

Le **sessioni** rappresentano uno strumento per gestire lo **stato della connessione client-server**, memorizzando dati temporanei.

Ogni sessione:

- Ha un identificativo univoco (**id di sessione**).
- Ha un insieme di **variabili di sessione**.

Implementazione di WIS con PHP

1. Una sessione viene creata dal server attraverso il metodo `start_session()`.
2. Viene inviato un cookie con l'id della sessione al browser del client.
3. Ad ogni richiesta da parte del browser per il medesimo sito, viene inviato il cookie per identificare la sessione corrente.
4. Per chiudere la sessione, il server puo' utilizzare il metodo `session_destroy()`.

Implementazione di WIS con PHP

In PHP, una sessione può avere delle **variabili temporanee associate**, memorizzate sul server.

Tutte le variabili di sessione sono entry di `$_SESSION`:

```
$_SESSION[ 'nomevariabile' ]= 'nuovovalore' ;
```

Per rimuovere una variabile, si usa il metodo `unset`:

```
unset($_SESSION[ 'nomevariabile' ]);
```

Implementazione di WIS con PHP

Nella pratica, **un programma PHP può essere composto da tanti file**. E' possibile utilizzare la direttiva `include` per includere un file sorgente (PHP/HTML) all'interno di un file PHP.

Utile per due motivi:

- **Suddividere un programma PHP in moduli**, eventualmente utilizzabili da altri moduli.
- **Suddividere la logica di gestione** dell'applicazione (in PHP) **dalla presentazione** dei dati (in HTML).

Implementazione di WIS con PHP

Esempio di **inclusione** di un file html all'interno di uno script PHP:

```
<?php
    if (isset($_POST['user']) and
isset($_POST['password']) {
        ... Codice PHP
    }
?>
include 'formlogin.html';
```

Implementazione di WIS con PHP

Il linguaggio PHP mette a disposizione una libreria molto vasta di **funzioni pre-definite** di vario tipo (operazioni su stringhe, file, mail, etc).

- **printf(\$formato, \$argomenti)**

Inserisce all'interno della stringa \$formato dei posizionatori tipizzati che verranno sostituiti dagli argomenti forniti come secondo argomento.

```
printf("Utente:%s Prodotti:%d",$user,$nrprod);
```

Implementazione di WIS con PHP

- **sprintf**(\$formato, \$argomenti)

Come la printf, ma **restituisce una stringa** in output.

```
$str=sprintf("Utente:%s Prodotti:%d",$user,$nrprod);
```

- **substr**(\$str,\$inizio,\$fine)

Estrae la sottostringa di \$str dal carattere \$inizio al carattere \$fine.

```
$str=substr("ProvaStringa",4,7);
```


Implementazione di WIS con PHP

- **str_replace**(\$str1, \$str2, \$str)

Sostituisce la stringa \$str1 con \$str2 all'interno della stringa \$str.

```
$str=str_replace("ab", "ac", "babbo");
```

- **split**(\$expr,\$str)

Suddivide una stringa \$str in base all'occorrenza di un separatore.

```
$array1=split("-", "335-6767690");
```

Implementazione di WIS con PHP

- **date(argomenti)**

Stampa la data corrente, in base alla formattazione definita dall'utente.

```
$echo date("m.d.y");
```

- **fopen, fread, fwrite, fclose, etc**

Operazioni su file (apertura, lettura, scrittura, etc)