Basi di Dati-X

Corso di Laurea in Informatica

Anno Accademico 2012/2013

Paolo Baldan

baldan@math.unipd.it

http://www.math.unipd.it/~baldan

Mantenimento dello stato

Mantenimento dello stato

- HTTP è un protocollo **stateless**

Mantenimento dello stato

- HTTP è un protocollo **stateless**
  - dopo la richiesta del client, il server invia la pagina e la connessione si chiude

Mantenimento dello stato

- HTTP è un protocollo **stateless**
  - dopo la richiesta del client, il server invia la pagina e la connessione si chiude
  - in PHP lo **scope delle variabili è locale** alla pagina

Mantenimento dello stato

3

- HTTP è un protocollo **stateless**
  - dopo la richiesta del client, il server invia la pagina e la connessione si chiude
  - in PHP lo **scope delle variabili è locale** alla pagina

- Come **mantenere lo stato** e passare dati tra pagine diverse di una applicazione web?

MySQL e PHP Corso di Basi di Dati
Saturday, June 1, 2013

Mantenimento dello stato

3

- HTTP è un protocollo **stateless**
  - dopo la richiesta del client, il server invia la pagina e la connessione si chiude
  - in PHP lo **scope delle variabili è locale** alla pagina
- Come **mantenere lo stato** e passare dati tra pagine diverse di una applicazione web?

  - talvolta è necessario passare pochi parametri da una pagina all'altra (Es. registrazione in più passi)

MySQL e PHP Corso di Basi di Dati
Saturday, June 1, 2013

Mantenimento dello stato

3

- HTTP è un protocollo **stateless**
  - dopo la richiesta del client, il server invia la pagina e la connessione si chiude
  - in PHP lo **scope delle variabili è locale** alla pagina
- Come **mantenere lo stato** e passare dati tra pagine diverse di una applicazione web?
  - talvolta è necessario passare pochi parametri da una pagina all'altra (Es. registrazione in più passi)

  - in altri casi il valore di alcune variabili deve essere accessibile da tutte le pagine che compongono l'applicazione (Es. carrello della spesa, identità dell'utente)

MySQL e PHP Corso di Basi di Dati
Saturday, June 1, 2013

Mantenimento dello stato

- Come mantenere lo stato e passare dati tra pagine diverse di una applicazione web?

Mantenimento dello stato

4

- Come mantenere lo stato e passare dati tra pagine diverse di una applicazione web?

- Da pagina a pagina
  - **Query string** (nella URL)
  - **Hidden fields** (delle form)

MySQL e PHP Corso di Basi di Dati
Saturday, June 1, 2013

Mantenimento dello stato

4

- Come mantenere lo stato e passare dati tra pagine diverse di una applicazione web?
- Da pagina a pagina
  - **Query string** (nella URL)
  - **Hidden fields** (delle form)

- Globali di applicazione o sessione
  - **Cookie** (memorizzati sul client)
  - **Variabili di sessione** (memorizzate sul server)
  - Memorizzazione sulla BD

MySQL e PHP Corso di Basi di Dati
Saturday, June 1, 2013

Query String

Saturday, June 1, 2013

Passaggio di parametri con query string

- Quando una form usa il metodo **GET**, i parametri sono codificati, come **query string**, nella URL che richiama la action
- La query string può essere "**costruita**" da programma
- **Esempio**:
  - prima pagina che offre una scelta tra vari studenti, matricola come link
  - seconda pagina che mostra i dati dell'utente selezionato

QueryString.php

Parametri con query string: Selezione


```
$query="SELECT Matricola, Nome, Cognome FROM Studenti";
$studenti=mysql_query($query,$conn);

/* fornisce in output i risultati in forma di tabella */
table_start(array("Matricola", "Nome", "Cognome"));

while ($row=mysql_fetch_row($studenti)) {
  /* trasforma il campo matricola in un link, che passi i tre
     parametri dello studente nella query string */
  $url=
    "QueryString1.php?matricola=" . urlencode($row[0]) .
                     "&nome="     . urlencode($row[1]) .
                     "&cognome="  . urlencode($row[2]);

  $row[0] = "<a href=\"$url\">" . $row[0] . "</a>";

  table_row($row);
};
```

QueryString.php

Parametri con query string: Selezione


```
$query="SELECT Matricola, Nome, Cognome FROM Studenti";
$studenti=mysql_query($query,$conn);

/* fornisce in output i risultati in forma di tabella */
table_start(array("Matricola", "Nome", "Cognome"));

while ($row=mysql_fetch_row($studenti)) {
  /* trasforma il campo matricola in un link, che passi i tre
     parametri dello studente nella query string */
  $url=
    "QueryString1.php?" .
     http_build_query(array('matricola' => $row[0],
                            'nome'      => $row[1],
                            'cognome'   => $row[2]));
  $row[0] = "<a href=\"$url\">" . $row[0] . "</a>";
  table_row($row);
};
```

QueryString.php

Parametri con query string: Visualizzazione


```
/* recupera i dati */
$matricola=$_GET['matricola'];
$nome=$_GET['nome'];
$cognome=$_GET['cognome'];

...

/* costruisce la query per recuperare gli esami */
$query=<<<END
 SELECT Data, Materia, Voto
        FROM Esami
        WHERE Candidato ="$matricola"
END;

/* la esegue e visualizza i risultati */
...
```

QueryString1.php

Reload problem

- Se l'informazione di operazione avvenuta con successo (es. per un inserimento nel database) è data dallo script che lo effettua, un reload causa un nuovo inserimento

Reload problem

10

- Se l'informazione di operazione avvenuta con successo (es. per un inserimento nel database) è data dallo script che lo effettua, un reload causa un nuovo inserimento

- Con POST la risottomissione di dati "vecchi" viene segnalata

`FormStudentiPOST.html`

Reload problem

10

- Se l'informazione di operazione avvenuta con successo (es. per un inserimento nel database) è data dallo script che lo effettua, un reload causa un nuovo inserimento

- Con POST la risottomissione di dati "vecchi" viene segnalata

`FormStudentiPOST.html`

- con GET no

`FormStudentiGET.html`

Reload problem

10

- Se l'informazione di operazione avvenuta con successo (es. per un inserimento nel database) è data dallo script che lo effettua, un reload causa un nuovo inserimento

- Con POST la risottomissione di dati "vecchi" viene segnalata

`FormStudentiPOST.html`

- con GET no

`FormStudentiGET.html`

- Soluzione: messaggio di successo in una pagina diversa

Saturday, June 1, 2013
Saturday, June 1, 2013
Saturday, June 1, 2013

Hidden Fields

Passaggio di parametri con hidden fields

- All'interno di una form possiamo specificare dei "fake input"

```
<input type="hidden" name="matricola" value="23456">
```

  - **Invisibili**, non richiedono input utente
  - Inviati assieme agli altri input quando si effettua submit

Saturday, June 1, 2013

Passaggio di parametri con hidden fields

12

- All'interno di una form possiamo specificare dei "fake input"

```
<input type="hidden" name="matricola" value="23456">
```

  - **Invisibili**, non richiedono input utente
  - Inviati assieme agli altri input quando si effettua submit

- Tipicamente pochi parametri per poche pagine

MySQL e PHP Corso di Basi di Dati

Saturday, June 1, 2013

Passaggio di parametri con hidden fields

12

- All'interno di una form possiamo specificare dei "fake input"

```
<input type="hidden" name="matricola" value="23456">
```

  - **Invisibili**, non richiedono input utente
  - Inviati assieme agli altri input quando si effettua submit

- Tipicamente pochi parametri per poche pagine

- Occorre creare una form in ogni pagina da cui il parametro deve essere passato

MySQL e PHP Corso di Basi di Dati

Saturday, June 1, 2013

Esempio con hidden fields

- **Esempio**: Wizards
  - prima pagina che recupera nome e cognome
  - seconda pagina che richiede l'indirizzo
  - terza pagina chiede la conferma e richiama il gestore dei dati

- La prima pagina è una normale form

```
<form method="get" action="Hidden.php">
   ...
   Nome:    <input type="text" name="nome" />
   Cognome: <input type="text" name="cognome" />
   <input type="submit" value="Avanti" />
   ...
</form>
```

Hidden.html

Esempio con hidden fields

- Seconda pagina

```
/* recupera i dati passati dalla pagina precedente */
$nome=$_REQUEST['nome'];
$cognome=$_REQUEST['cognome'];

/* e richiede i nuovi */
echo<<<END
<form method="get" action="Hidden2.php">
  ...
  <!-- Hidden fields Nome e Cognome -->
  <input type="hidden" name="nome" value="$nome">
  <input type="hidden" name="cognome" value="$cognome">

  Indirizzo:    <input type="text" name="indirizzo" />
  Citt&agrave;: <input type="text" name="citta" />
  ...
</form>
END;
```

Hidden1.php

Esempio con hidden fields

- Terza pagina

```
/* recupera i dati passati dalla pagina precedente */
$nome=$_REQUEST['nome'];
$cognome=$_REQUEST['cognome'];
$indirizzo=$_REQUEST['indirizzo'];
$citta=$_REQUEST['citta'];

/* li mostra */
echo "<h4>Prospetto riassuntivo</h4>";

echo "
<ul>
  <li>Nome: $nome</li>
  <li>Cognome: $cognome</li>
  <li>Indirizzo: $indirizzo</li>
  <li>Citta: $citta</li>
</ul>";
```

Hidden2.php

Esempio con hidden fields

- Terza pagina

```
/* e chiede conferma, passando quindi il controllo
   al gestore */

echo<<<END
<form method="GET" action="Hidden3.php">
  <!-- Hidden fields -->
  <input type="hidden" name="nome" value="$nome">
  <input type="hidden" name="cognome" value="$cognome">
  <input type="hidden" name="indirizzo" value ="$indirizzo">
  <input type="hidden" name="citta" value="$citta">
  <input type="submit" value="Conferma">
</form>
END;
```

Hidden2.php

Detour - Gestione dei BLOB - inserimento


```
/* path del file da memorizzare */
$file="image.jpg";

/* apre il file (r = read, b = binary) */
$fp = fopen($img, "rb");
/* lo mette in una variabile e quota caratteri
   problematici per MySQL (Es. ‘ e “) */
$content = fread($fp, filesize($img));
$content = mysql_real_escape_string($content, $conn);

/* costruisce la query per memorizzare il file come blob
   nella tabella BlobTable(Id, File) e la esegue */
$fileId="01";
$query="INSERT INTO BlobTable(Id, File)
        VALUES (\"$fileId\", \"$content\")";

mysql_query($query, $conn);
```


Gestione dei BLOB - estrazione


```
/* recupera il file dalla tabella */
$query="SELECT File FROM BlobTable
        WHERE Id=\"$fileId\"";

/* Esegue la query */
$result = mysql_query($query, $conn);

/* recupera il risultato e lo fornisce al browser,
   segnalando, tramite header, che e` una immagine */
$recover = mysql_fetch_array($result);
header("Content-type: ". $mime);
echo $recover['File'];
```


Cookies

Cookies

- Meccanismo che permette al server di memorizzare e quindi reperire **piccole quantità di dati sul client**
  - info sulla visita
  - identità del visitatore, ecc.
- Es.: i siti che “si ricordano di noi” quando ritorniamo
- In pratica:
  - inviati dal server al client
  - memorizzati in un file/db, nel file system del client
  - rimandati al server ad ogni visita successiva da parte del client

Cookies

- Un cookie è
  - **temporaneo**: la durata è configurabile e tipicamente si estende per molte visite del sito
  - **piccolo**: tipicamente max 4 kb
- Problemi di privacy
- I cookie possono essere disattivati (ma questo può determinare il mancato funzionamento di applicazioni web)

Creazione di cookies

- Un cookie può essere creato con:

```
setcookie(name, value, expire, path, domain)
```

  - **name** e **value**: nome del cookie e valore associato
  - **expire**: tempo di scadenza (in secondi, Unix time)
  - **path**: directory di validità sul server (es: `"/"` , `"/Eser4"`)
  - **domain**: dominio di validità (es. `"unipd.it"`)
- Da usare prima di qualsiasi output!

Accesso ai cookies

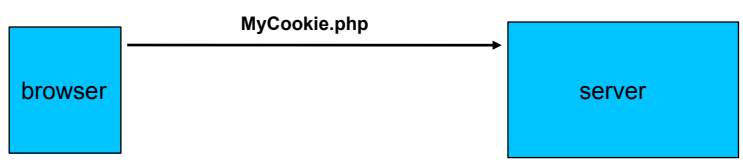
- Ai cookie si accede tramite il superglobal array **$_COOKIE**

  $_COOKIE["name"]
- Un cookie settato in una pagina è **visibile solo al caricamento successivo** (i cookie vengono inviati nello header HTTP)

Funzionamento dei cookie

MyCookie.php

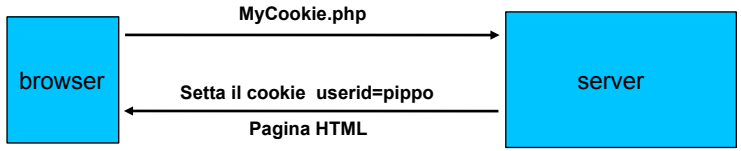
```
<?php setcookie('userid','pippo'); ?>
<html><head>...</head>
<body> Valore del cookie: <?php echo $_COOKIES['userId']"; ?>
</body>
</html>
```



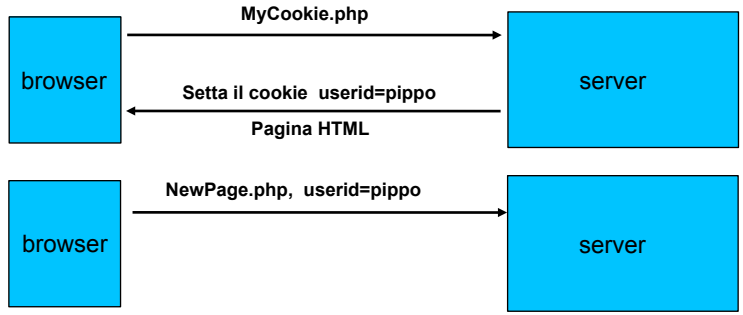
Funzionamento dei cookie


```
<?php setcookie('userid','pippo'); ?>
<html><head>...</head>
<body> Valore del cookie: <?php echo $_COOKIES['userId']"; ?>
</body>
</html>
```

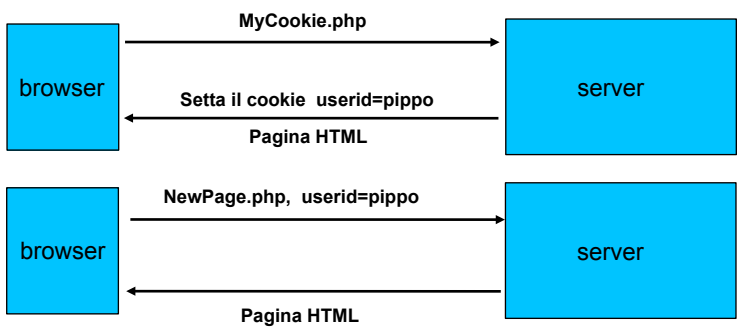
MyCookie.php



Funzionamento dei cookie
24
<?php setcookie('userid','pippo'); ?>
<html><head>...</head>
<body> Valore del cookie: <?php echo $_COOKIES['userId']"; ?>
</body>
</html>
MyCookie.php



Funzionamento dei cookie
24
<?php setcookie('userid','pippo'); ?>
<html><head>...</head>
<body> Valore del cookie: <?php echo $_COOKIES['userId']"; ?>
</body>
</html>
MyCookie.php



Cookies: Esempio


```
/* Conta il numero di visite */
$count=$_COOKIE["visite"];        /* prova a recuperare
                                     il cookie */
if (isset($count))
  $count++;
else
  $count=0;

/* nuovo cookie (durata un'ora, validità intero dominio.) */
setcookie("visite", "$count", time()+3600, "/");

/* L'output puo` cominciare */
...
if ($count)
  echo "Sei stato qui $visite volte!";
else
  echo "Non sei mai stato qui!";
...
```

cookie-basic.php

Eliminazione dei cookie

- Usare la funzione setcookie
  - per cancellare il valore

    setcookie("visite")
  - o per assegnare un tempo di scadenza già passato:

```
setcookie("visite", "", time()-3600, "/")
```

- Ad esclusione del valore, un cookie deve essere cancellato con gli **stessi parametri di creazione!**

Esempio: Personalizzazione di una pagina

- Supponiamo di volere permettere ad un utente di personalizzare la pagina di accesso ad un sito con
  - messaggio di benvenuto
  - colore della pagina
- Tre pagine:
  - **cookie-pers.php**: pagina principale
  - **cookie-pers-form.php**: pagina con la form per il setting dei dati
  - **cookie-pers-set.php**: pagina che registra le modifiche

Esempio: cookie-pers.php


```
/* Verifica se il cookie con i setting e` presente */
$colore = $_COOKIE["colore_cookie"];
$nome   = $_COOKIE["nome_cookie"];

/* valore standard per il colore */
if (! $colore)
  $colore="Azure";
echo "<body bgcolor=\"$colore\">";

/* messaggio personalizzato */
if ($nome)
  $msg="Buongiorno $nome! Bentornato nel nostro sito";
else
  $msg="Buongiorno!";

echo "<h2>ACME Corporation</h2>";
echo "<h3>$msg</h3>";
...
echo "<a href=\"cookie-pers-form.php\">Cambia impostaz.</a>";
```

cookie-pers.php

Esempio: cookie-pers-form.php


```
/* recupera il valore eventualmente gia' settato per i cookie */
$colore=$_COOKIE["colore_cookie"];
$nome=$_COOKIE["nome_cookie"];

/* colori e relative labels */
$set_colori=array("-" => "Default",
                  "Yellow" => "Giallo",
                  "Azure" => "Azzurro",
                  "Pink" => "Rosa",
                  "Snow" => "Neve");

/* presenta la form di personalizzazione ... */
...
```

Esempio: cookie-pers-form.php


```
/* presenta la form di personalizzazione ... */
echo<<<END
<form method="get" action="cookie-pers-set.php">
  Nome:   <input type="text" name="nome" value="$nome">
  Colore: <select name="colore">
END;
/* se $colore è settato lo presenta come default */
if ($colore)
  echo "<option value=\"$colore\">$set_colori[$colore]</option>";
/* ... poi presenta gli altri */
foreach ($set_colori as $key => $lab) {
  if ($key!=$colore)
    echo "<option value=\"$key\">$lab</option>";
};
echo<<<END
</select>
<input type="submit" value="Set">
</form>
END;
```


Esempio: cookie-pers-set.php


```
/* recupera i dati passati dalla form */
$nome=trim($_REQUEST["nome"]);
$colore=$_REQUEST["colore"];

/* setta il nuovo valore dei cookie (15 gg) o li cancella */
if ($nome)
  setcookie("nome_cookie", "$nome", time()+3600*24*15);
else
  setcookie("nome_cookie");

if ($colore)
  setcookie("colore_cookie", "$colore", time()+3600*24*15);
else
  setcookie("colore_cookie");

/* L'output puo` cominciare */
echo<<<END
<html>
...
```


Riassunto

- Mantenimento dello stato
  - passaggio di parametri tra pagine successive
    - query string
    - hidden fields
  - stato dell'applicazione
    - cookies
    - sessioni

Sessioni

Sessioni

- Meccanismo per il mantenimento dello stato dell'applicazione web in pagine distinte

Sessioni

34

- Meccanismo per il mantenimento dello stato dell'applicazione web in pagine distinte

  - Alla **prima richiesta** da parte del browser di una risorsa, viene creata una sessione con un **session id (sid)**, ritornato al client [cookie!]

MySQL e PHP Corso di Basi di Dati
Saturday, June 1, 2013

Sessioni

34

- Meccanismo per il mantenimento dello stato dell'applicazione web in pagine distinte
  - Alla **prima richiesta** da parte del browser di una risorsa, viene creata una sessione con un **session id (sid)**, ritornato al client [cookie!]

  - Alla sessione sono associate **variabili di sessione** memorizzate sul server HTTP (e lì stanno! ... piu` sicure di query string, hidden fields, cookies !!!)

MySQL e PHP Corso di Basi di Dati
Saturday, June 1, 2013

Sessioni

34

- Meccanismo per il mantenimento dello stato dell'applicazione web in pagine distinte
  - Alla **prima richiesta** da parte del browser di una risorsa, viene creata una sessione con un **session id (sid)**, ritornato al client [cookie!]
  - Alla sessione sono associate **variabili di sessione** memorizzate sul server HTTP (e lì stanno! ... piu` sicure di query string, hidden fields, cookies !!!)

  - Nelle **richieste successive**, il browser fornisce il **session id** per identificare la sessione; quindi il server può recuperare le variabili di sessione relative.

MySQL e PHP Corso di Basi di Dati
Saturday, June 1, 2013

Sessioni

- Meccanismo per il mantenimento dello stato dell'applicazione web in pagine distinte
  - Alla **prima richiesta** da parte del browser di una risorsa, viene creata una sessione con un **session id (sid)**, ritornato al client [cookie!]
  - Alla sessione sono associate **variabili di sessione** memorizzate sul server HTTP (e lì stanno! ... piu` sicure di query string, hidden fields, cookies !!!)
  - Nelle **richieste successive**, il browser fornisce il **session id** per identificare la sessione; quindi il server può recuperare le variabili di sessione relative.

Sessioni

34

- Meccanismo per il mantenimento dello stato dell'applicazione web in pagine distinte
  - Alla **prima richiesta** da parte del browser di una risorsa, viene creata una sessione con un **session id (sid)**, ritornato al client [cookie!]
  - Alla sessione sono associate **variabili di sessione** memorizzate sul server HTTP (e lì stanno! ... piu` sicure di query string, hidden fields, cookies !!!)
  - Nelle **richieste successive**, il browser fornisce il **session id** per identificare la sessione; quindi il server può recuperare le variabili di sessione relative.
  - La sessione dura fino a che non viene **esplicitamente terminata**, si **chiude il browser**, oppure la **sessione scade**.

MySQL e PHP Corso di Basi di Dati
Saturday, June 1, 2013

Sessioni: session_start()

- Ogni script che accede ai dati della sessione **inizia** con

```
session_start()
```


Sessioni: session_start()

35

- Ogni script che accede ai dati della sessione **inizia** con

```
session_start()
```
  - se la richiesta HTTP non ha associato un **session id** (**sid**), il server ne crea uno nuovo e lo restituisce al browser

MySQL e PHP Corso di Basi di Dati
Saturday, June 1, 2013

Sessioni: session_start()

- Ogni script che accede ai dati della sessione **inizia** con

  `session_start()`

  - se la richiesta HTTP non ha associato un **session id** (**sid**), il server ne crea uno nuovo e lo restituisce al browser
  - se invece ha **sid**, il server recupera le variabili associate alla sessione e le rende disponibili in **`$_SESSION`**

35
Sessioni: session_start()
Ogni script che accede ai dati della sessione inizia con
session_start()
se la richiesta HTTP non ha associato un session id (sid), il server ne crea uno nuovo e lo restituisce al browser
se invece ha sid, il server recupera le variabili associate alla sessione e le rende disponibili in $_SESSION

  - lo script può creare **variabili di sessione**, memorizzate in **`$_SESSION`**

MySQL e PHP Corso di Basi di Dati
Saturday, June 1, 2013
35
Sessioni: session_start()
Ogni script che accede ai dati della sessione inizia con
session_start()
se la richiesta HTTP non ha associato un session id (sid), il server ne crea uno nuovo e lo restituisce al browser
se invece ha sid, il server recupera le variabili associate alla sessione e le rende disponibili in $_SESSION
lo script può creare variabili di sessione, memorizzate in $_SESSION

  - la creazione di una sessione deve avvenire a livello root dell'applicazione, altrimenti risulta inaccessibile da alcune pagine (cookie)

MySQL e PHP Corso di Basi di Dati
Saturday, June 1, 2013
35
Sessioni: session_start()
Ogni script che accede ai dati della sessione inizia con
session_start()
se la richiesta HTTP non ha associato un session id (sid), il server ne crea uno nuovo e lo restituisce al browser
se invece ha sid, il server recupera le variabili associate alla sessione e le rende disponibili in $_SESSION
lo script può creare variabili di sessione, memorizzate in $_SESSION
la creazione di una sessione deve avvenire a livello root dell'applicazione, altrimenti risulta inaccessibile da alcune pagine (cookie)

  - la sessione termina quando si chiude il browser / è distrutta / scade

MySQL e PHP Corso di Basi di Dati
Saturday, June 1, 2013

Sessioni: gestione dei dati

- Per creare una variabile di sessione

```
$_SESSION["var"] = value
```

- e per distruggerla

```
unset($_SESSION["var"])
```

- Per l'unset di tutte le variabili della sessione

```
$_SESSION = array()
```


Sessioni: session_destroy()

- Per "azzerare" i dati associati ad una sessione

```
session_destroy()
```

- NOTA:
  - non esegue l'unset delle variabili associate alla sessione (sono quindi ancora accessibili nella pagina corrente)
  - non elimina il cookie associato alla sessione; occorre farlo esplicitamente

```
$sname=session_name();
if (isset($_COOKIE[$sname])) {
    $par = session_get_cookie_params();
    setcookie($sname, '', time() - 42000,
        $par["path"], $par["domain"],
        $par["secure"], $par["httponly"]
    );}
```


Esempio: Wizard con sessioni

- Wizard che riceve l'input dell'utente in due fasi
  - prima pagina: nome e cognome
  - seconda pagina: indirizzo e città
  - terza pagina conferma
- permettendo di navigare all'interno delle pagine, mantenendo i dati specificati

session-form-0.php

Esempio: session-form-0.php (prima pagina)


```
/* inizia o attiva la sessione */
session_start();

/* recupera nome e cognome eventualm. gia' specificati */
$nome=$_SESSION['nome'];
$cognome=$_SESSION['cognome'];

/* form */
echo<<<END
/* html ... */
<form method="get" action="session-form-check-0.php" />
  Nome    <input type="text" name="nome" value="$nome" />
  Cognome <input type="text" name="cognome" value="$cognome" />
  <input type="submit" name="submit" value="Avanti" />
  <input type="submit" name="cancel" value="Cancella tutto" />
</form>
...
END;
```

session-form-0.php

Header

- Quando risponde ad una richiesta, il server web invia un response HTTP header standard
  - **Status line**

    HTTP/1.0 200 OK oppure

    HTTP/1.0 404 Not Found
  - **Headers**

```
Date: Fri, 31 Dec 2009 23:59:59 GMT
Content-Type: text/html
Content-Length: 1354
...
<html>
  <body>
  ...
```


Header

- PHP permette inviare degli header HTTP **prima di ogni output** con il comando

  header(string)
- **Esempio**
  - Ridirezione ad altra pagina

```
<?php
header('Location: http://www.math.unipd.it');
?>
```

  - Invia lo status di pagina non trovata

```
<?php
header("HTTP/1.0 404 Not Found");
?>
<html> <body>Pagina non trovata!</body></html>
```


Header

- Download di un file:

```
/* Mime-type del documento */
header("Content-type:application/pdf");

/* Il file sarà downloaded.pdf */
header("Content-Disposition:attachment;
        filename='download.pdf'");

/* Il file da inviare è in original.pdf */
readfile("original.pdf");
...
```


Esempio: session-form-check-0.php


```
/* attiva la sessione */
session_start();

/* url base dello script corrente */
$base=base_url();

if ($_GET['cancel']) {
  /* se si e' arrivati premendo 'cancel' */
  $_SESSION=array();
  header("Location: $base/session-form-0.php");
} else {
  /* altrimenti, se si e` arrivati premendo 'submit',
     registra i dati e passa alla form successiva */
  $_SESSION['nome']=$_GET['nome'];
  $_SESSION['cognome']=$_GET['cognome'];
  header("Location: $base/session-form-1.php");
```

Esempio: funzione base_url


```
/* ritorna l'url base dello script corrente */
function base_url() {
  return "http://{$_SERVER['HTTP_HOST']}"
         . dirname($_SERVER['PHP_SELF']);
};
```


Esempio: session-form-3.php

- Per la pagina con indirizzo e città e la pagina di conferma, gli script sono analoghi.
- Lo script richiamato dall'ultima pagina (conferma) distrugge la sessione:

```
/* distrugge la sessione */
session_start();

session_destroy();
$_SESSION=array();
$sname=session_name();
if (isset($_COOKIE[$sname]))
   $par = session_get_cookie_params();
   setcookie($sname, '', time() - 42000,
         $par["path"], $par["domain"],
         $par["secure"], $par["httponly"]
};
```


Sessioni multiple

- È possibile avere più sessioni attive, attribuendo a queste un nome

```
session_name("name");
```

  - cambia il nome della sessione corrente e restituisce il precedente
  - se `"name"` è vuoto o non specificato, mantiene il nome precedente

- Per creare una sessione con il nome `"name"` occorre chiamare la funzione prima di `session_start()`
- Se si usa una sessione unica, basta tenere il nome di default

Esempio: Login multipli - form


```
/* Form di login (gestore single session) */

<h3>Single Session</h3>

<form method="get" action="SingleSession.php">
  <fieldset>
    <legend>Login</legend>

    Username: <input type="text" name="user" />
    <input type="submit" value="Login" />
  </fieldset>
</form>
```

LoginSingleSession.html

Esempio: Login multipli - gestore single session


```php
<?php
/* Recupera il nome dell'utente */
$user=$_GET["user"];

/* inizia la sessione */
session_start();

/* conta il numero delle visite */
if (empty($_SESSION['count'])) {
  $_SESSION['count'] = 1;
} else {
  $_SESSION['count']++;
}
```

SingleSession.php

Esempio: Login multipli - gestore single session


```php
/* e lo fornisce in output */
$count=$_SESSION['count'];
echo<<<END
...
<h3>Single Session</h3>

<p>
Utente: $user  <br />
Numero Visite: $count
</p>
</html>
END;

?>
```

SingleSession.php

Esempio: Login multipli - form


```php
/* Form di login (gestore multi session) */

<h3>Single Session</h3>

<form method="get" action="MultiSessions.php">
  <fieldset>
    <legend>Login</legend>

    Username: <input type="text" name="user" />
    <input type="submit" value="Login" />
  </fieldset>
</form>
```

LoginMultiSessions.html

Esempio: Login multipli - gestore single session


```php
<?php
/* Recupera il nome dell'utente */
$user=$_GET["user"];

/* inizia la sessione */
session_name("$user");
session_start();

/* conta il numero delle visite */
if (empty($_SESSION['count'])) {
  $_SESSION['count'] = 1;
} else {
  $_SESSION['count']++;
}
```

MultiSessions.php

Esempio: Login multipli - gestore single session


```
/* e lo fornisce in output */
$count=$_SESSION['count'];
echo<<<END
...
<h3>Single Session</h3>

<p>
Utente: $user  <br />
Numero Visite: $count
</p>
</html>
END;

?>
```

MultiSessions.php

Sessioni senza cookie

- È possibile utilizzare le sessioni senza i cookie attivati
- In questo caso occorre passare il sid tra pagina e pagina
  - in una url usa la costante `SID` (`session_name()=session_id()`)

```
$str = htmlspecialchars(SID);
<a href="EsempioSID.php?$str">click here</a>
```

  - in una form, mediante un campo hidden

```
$sname=session_name();
$sid=session_id();
<input type="hidden" name="$sname" value="$sid">
```

- il ricevente lo recupera automaticamente

Cookie, Sessioni, sicurezza

- E` troppo chiedere all'utente di attivare i cookies?
  - No, ma ricordare come funzionano ....
- Sono memorizzati sul client e passati tramite http
  - evitare informazioni sensibili in chiaro
  - non utilizzare cookie costruibili "artificialmente" per abilitare operazioni delicate (es. "username logged" no!)
  - https e secure cookies
  - usare "session-like" cookie che identifichino l'utente, lasciando le informazioni sensibili sul server

Autenticazione

Autenticazione

- Uso delle **sessioni** o dei **cookie**
- **Autenticazione HTTP**
- **Password hard-coded** nel source o **memorizzate nel database** SQL

Autenticazione con sessioni

- **Idea**:
  - Quando l'utente effettua il **login** fornisce le proprie credenziali
  - Se passa il controllo vengono settate opportune variabili di sessione
  - All'inizio di ogni script ad accesso ristretto si controlla che le variabili di interesse siano settate
- Le password possono essere
  - hard-coded nel sorgente
  - memorizzate in una tabella nel DB
- Per ragioni di sicurezza le password sono crittate (si memorizza un hash)

Esempio: form di login


```
<form method="post" action="session-auth.php" />
...
Login:    <input type="text"     name="login" />
Password: <input type="password" name="password"
                maxlength="8" />
<input type="submit" value="Vai">

Devi <a href="session-auth-register.php">registrarti</a>?
...

</form>
```

session-auth-login.php

Esempio: gestore del form di login


```
/* recupera i dati immessi */
$login=$_POST['login'];
$password=$_POST['password'];

/* verifica se login e' valido e recupera la password */
$db_pwd=get_pwd($login);

if ($db_pwd && (SHA1($password) == $db_pwd)) {
  /* se login e' valido e la password e' corretta ...
     registra i dati nella sessione */
  session_start();
  $_SESSION['login']=$login;
  ...
} else {
  /* se login e' invalido o la password e' incorretta, rimanda
     alla pagina di login */;
  ...
};
```

session-auth-login-manage.php

Esempio: una pagina generica


```
/* attiva la sessione */
session_start();

/* verifica se la variabile 'login' e` settata */
$login=$_SESSION['login'];

if (empty($login)) {
  /* non passato per il login: accesso non autorizzato ! */
  echo "Impossibile accedere. Prima effettuare il login.";

} else {
  /* accesso autorizzato     */
  /* - normali operazioni    */
  ...
  /* - possibilità di logout */
  ...
};
```

session-auth-operate.php

Esempio: logout


```
/* attiva la sessione */
session_start();

/* sessione attiva, la distrugge */
$sname=session_name();

session_destroy();

/* ed elimina il cookie corrispondente */
if (isset($_COOKIE[$sname])) {
  setcookie($sname,'', time()-3600,'/');
};

echo "Logout effettuato <b>" . $_SESSION['login'] . "</b>!";
```

session-auth-logout.php

Esempio: altre note

- Occorre prevedere anche
  - una pagina di registrazione con l'immissione di
    - login, password, conferma della password
  - il gestore della pagina di registrazione che controlla
    - se password=conferma
    - se login è già in uso
    - memorizza le informazioni in una tabella SQL
    - della password si memorizza un hash

Esempio: gestore della registrazione


```
/* recupera i dati immessi */
$login=$_POST['login'];
$password=$_POST['password'];
$confirm=$_POST['confirm'];

/* Verifica se login e password soddisfano opportuni
   criteri ... alfanumerici, lunghezza, ecc. */
...
/* password e conferma sono uguali? login è in uso? */
if ($password != $confirm)
  echo "Errore! Password e Conferma sono diverse. ";
elseif (get_pwd($login)) {
  echo "Errore! Login gia` in uso. ";
} else {
  /* inserisce il login e password nella BD */
  new_user($login, $password);
  echo "Registrazione effettuata con successo!";
};
```

Autenticazione con cookies

- Stesse idee dell'autenticazione con sessioni
  - Quando l'utente effettua il login fornisce le proprie credenziali
  - Se passa il controllo vengono settato un opportuno cookie
  - All'inizio di ogni script ad accesso ristretto si controlla che il cookie esista (e contenga le info corrette)
  - il logout elimina il cookie
- **Nota**: Il login sopravvive alla chiusura del browser, per un tempo configurabile

Autenticazione con cookies

- Cosa inserire nel cookie?
  - non "username logged" ...
  - username con un segreto verificabile nel server.
- Contenuto:
  - username + password (hash ... possibile, ma inutilmente pericoloso)
  - username + hash (username + parola segreta [fissata])
  - username + hash (username + timestamp [memorizzato])
  - ....

Esempio: gestore del form di login


```
/* recupera i dati immessi */
$login=$_POST['login'];
$password=$_POST['password'];

/* verifica se login e' valido e recupera la password */
$db_pwd=get_pwd($login);

if ($db_pwd && (SHA1($password) == $db_pwd)) {
  /* se login e' valido e la password e' corretta ...
     registra i dati nella sessione */
  $hash = SHA1($login . $secret);
  setcookie('login', $login . "," . $hash, ...);
  ...
} else {
  /* se login e' invalido o la password e' incorretta, rimanda
     alla pagina di login */;
  ...
};
```

session-auth-login-manage.php

Esempio: una pagina generica


```
/* recupera il cookie */
list($login,$hash)= split(',', $COOKIE['login']);


/* verifica la correttezza del cookie */

if ( SHA1($login . $secret) != $hash  ) {
  /* non passato per il login: accesso non autorizzato ! */
  echo "Impossibile accedere. Prima effettuare il login.";

} else {
  /* accesso autorizzato     */
  /* - normali operazioni    */
  ...
  /* - possibilità di logout */
  ...
};
```

session-auth-operate.php

Autenticazione HTTP

- Il web-server può richiedere autenticazione per l'accesso ad alcune risorse
  - Se l'utente richiede una risorsa "protetta" (`GET url ...`)
  - Il server risponde dicendo è necessaria l'autenticazione

```
HTTP/1.0 401 Authorization Required
...
WWW-Authenticate: Basic realm="MyRealm"
```

http-auth.php

  - Il client fa una seconda richiesta inviando username/password

```
GET url ...
Authorization: Basic username/password
```

  - Se sono corretti, si accede alla risorsa
  - Ad ogni accesso successivo il browser invia le credenziali ...

Autenticazione HTTP in PHP

- PHP può utilizzare il meccanismo di autenticazione fornito da HTTP
- Invia lo header HTTP **WWW-Authenticate**

```
header('WWW-Authenticate: Basic realm="Messaggio"');
```

fa apparire una finestra, con intestazione "**Messaggio**" che richiede che l'utente fornisca Username/Password

- I valori forniti sono inviati ad ogni accesso allo **stesso dominio** (memorizzati nella **cache del browser**) e accessibili come variabili di connessione

```
$_SERVER['PHP_AUTH_USER']
$_SERVER['PHP_AUTH_PW']
```


Autenticazione HTTP

- **Idea**:
  - login alla radice dell'area protetta
  - quando l'utente accede ad uno script protetto, si controlla se sono settate e se hanno un valore corretto le variabili di connessione `$_SERVER['PHP_AUTH_USER'], $_SERVER['PHP_AUTH_PW']`
  - in caso affermativo si dà accesso alla pagina
  - in caso negativo, si fornisce un messaggio di errore o si rimanda alla pagina di login

Esempio: funzione di autenticazione


```
function authenticate() {
  $user=$_SERVER['PHP_AUTH_USER'];
  $pwd= $_SERVER['PHP_AUTH_PW'];

  if (! check($user,$pwd)) {
    /* informazioni assenti o scorrette ->  finestra di login */
    Header("WWW-Authenticate: Basic realm=\"Area Riservata\"");
    Header("HTTP/1.0 401 Unauthorised");
    $auth=FALSE;
  } else {
    /* utente autenticato */
    $auth=TRUE;
  };
  return $auth;
}
```

Esempio: Script di login

- Login

```
authenticate()
  or die("Accesso negato!");

echo "Login effettuato!";
```

http-auth-login.php

Esempio: Script

- Script in area riservata

```
$user=$_SERVER['PHP_AUTH_USER'];
$pwd= $_SERVER['PHP_AUTH_PW'];


check($user,$pwd) or
  die("Accesso negato! Torna al
       <a href=\"http-auth-login.php\">login</a>");


/* pagina di operazione */
echo "Bene! Eri autenticato!";
```

http-auth-operate.php

Esempio: Script

- Può essere sensato anche permettere l'autenticazione in ogni pagina ... dipende dall'applicazione

```
authenticate() or
  die("Accesso negato!");


/* pagina di operazione */
echo "Bene! Eri o ti sei autenticato!";
```

http-auth-operate.php

Object orientation

PHP e gli oggetti

- Il modello a oggetti di PHP prevede
  - classi e oggetti (istanze di classe)
  - incapsulamento (attributi e metodi private, protected, public)
  - ereditarietà (extend)
  - overriding e controllo sull'overriding (final)
  - classi astratte e interfacce

Classi: attributi e metodi


```
class User {
  /* attributi (proprietà) di istanza */
  private $id;
  protected $name;
  protected $tel;

  /* costruttore e distruttore */
  public function __construct($id, $name, $tel="") {
    $this->id = $id;
    $this->name = $name;
    $this->tel = $tel;
  }

  public function __destruct() {
    /* chiude file, dealloca risorse */
    ...
  }
```


Classi: attributi e metodi


```
  /* getter e setter ... */

  public function getTel() {
    return $this->tel;
  }

  public function setTel($tel) {
    $this->tel = $tel;
  }

  /* altri metodi ... */

}
```


Visibilità

- Proprietà e metodi possono essere:
  - private
    visibile solo all'interno della classe
  - protected
    visibile dalle sottoclassi (e superclasse)
  - public
    visibile a tutti

Creazione e accesso


```
/* Creazione e accesso */
$user1 = new User(4, "Pippo", "047-22121");

echo $user1->getTel();      /* out: 047-22121 */

$user->setTel("020-20021");


/* costruttore con il valore di default per tel */
$user2 = new User(6, "Pluto");
```


Costanti, proprietà e metodi di classe


```
class User {
  /* costanti e attributi (proprieta') di classe */
  const maxUsers = 10;
  protected static $numUsers = 0;

  public static function getNumUsers() {
    return self::$numUsers;
  }

  /* costruttore e distruttore */
  public function __construct($id, $name, $tel="") {
    self::$numUsers++;
        ...
  }

  public function __destruct() {
    self::$numUsers--;
  }
```


Operatore di scope resolution

- Per accedere alle proprietà di classe con il nome della classe:

```
echo User::getNumUsers();
echo User::maxUsers;
```

- Nei metodi si possono usare
  - **self**
    classe dell'oggetto invocante
  - **parent**
    classe genitore dell'oggetto invocante

Sottoclassi


```
class Student extends User {

  /* attributo proprio */
  private $year;

  /* overriding del costruttore */
  public function __construct ($id, $name, $tel="", $year=NULL)
  {
    parent::__construct($id, $name, $tel);
    if ($year)
      $this->year = $year;
    else
      $this->year=date('Y');
  }

  /* metodi propri e overriding di metodi della superclasse */

}
```

Sottoclassi: final


```
class Student extends User {

  ...

  /* metodo final: impossibile fare overriding
     in sottoclassi */
  final public function setVote() {

}
```


Classes: Autoloading

- È comune organizzare il codice con una classe per file
- magic method **__autoload** permette di evitare di includere esplicitamente i file usati
- Richiamato quando si usa una classe non definita

```
<?php
function __autoload($class_name) {
    require_once $class_name . '.php';
}
```

- Assume che la classe **$class_name** si trovi nel file **$class_name.php**

Classi astratte

- Una classe astratta, con un metodo save istanziato nelle sottoclassi

```
abstract class Data {
  protected $data;  /* some data */

  /* costruttore */
  public function __construct ($data) {
    $this->data = $data;
  }

  /* metodo astratto */
  abstract public function save ($name);
}
```


Classi astratte


```
class FileData extends Data {

  /* istanzia il metodo save */
  public function save ($name) {
    $file = fopen($name . ".txt", "w");
    fwrite($file, $this->data);
  }
}


class CokieData extends Data {

 /* istanzia il metodo save */
  public function save ($name) {
    setcookie($name, $this->data);
  }
}
```

Interfacce

- Una classe può implementare varie interfacce (ed estendere una sola classe, anche astratta)

```
interface Iterator {
  /* ritorna l'elemento corrente */
  public function current ();
  /* ritorna la chiave corrente (scalare) */
  public function key ();
  /* va al prossimo elemento */
  public function next ();
  /* va all'inizio */
  public function rewind ();
  /* la posizione corrente e` valida? */
  public function valid ();

}
```


Interfacce: Esempio

- Classe che implementa iterator:

```
class Tre implements Iterator {
  /* dati */
  protected $val;  /* array dei valori */
  protected $pos;  /* posizione corrente */

  /* costruttore */
  public function __construct ($primo, $secondo, $terzo) {
    $this->val =  array ($primo, $secondo, $terzo);
    $this->pos = 0;
  }
```


Interfacce: Esempio


```
/* elemento corrente */
public function current () {
  return $this->val[$this->pos];
}

/* chiave corrente */
public function key () {
  return $this->pos;
}

/* prossimo elemento */
public function next () {
  $this->pos++;
}
```

```
  /* va all'inizio */
  public function rewind () {
    $this->pos = 0;
  }

  /* pos. corrente valida? */
  public function valid () {
    return ($this->pos <3);
  }
}
```


Interfacce: Esempio

- Poi si puo' usare foreach per iterare ....

```
$a = new tre('a', 'b', 'c');

foreach ($a as $val) {
  echo "valore: $val\n";
}
```

```
Output:

valore: a
valore: b
valore: c
```

Magic method __toString

- chiamato quando si usa l'oggetto come una stringa

```
/* nella classe User ... */
public function __toString () {
   $str= "Id: $this->id\n" .
     "Name: $this->name\n" .
     "Tel: $this->tel\n";
   return $str;
 }
```

```
/* quindi ... */
$user = new User(...);
echo $user;  /* richiama toString */
```


Magic methods __get, __put

- `__get($attr)`

  chiamato quando si accede in lettura a proprietà inaccessibili, `$attr` contiene il nome della proprietà

- `__set($attr, $val)`:

  chiamato quando si accede in scrittura a proprietà inaccessibili, `$attr` contiene il nome della proprietà e `$val` il valore

Magic methods __get, __put: Esempio

- Getter e setter universali

```
public function __get ($attr) {
  if (isset($this->$name))
     return $this->$name;
}

public function __set ($attr,$val) {
  if (isset($this->$name))
      $this->$name=$val;
}
```

```
/* in User ... */
$user->tel = "042-353535"
/* richiama __set("tel", "042-353535") */

echo $user->tel;
/* richiama __get("tel") */
```


Ecc. ecc. ecc.

- Eccezioni

```
function div($x,$y) {
  if (! $y)
    throw new Exception('Divisione per 0');
  else
    return $x/$y;
};


try {
  echo div(5,2) . "\n";
  echo div(5,0) . "\n";
} catch (Exception $e) {
  echo 'Caught exception: ',  $e->getMessage(), "\n";
}
```

PDO

- PHP database object

```
/* parametri per la connessione */
$dbhost="127.0.0.1";     /* server MySQL */
$dbuser="";              /* utente       */
$dbpwd="";               /* password     */
$dbname="Universita";    /* database da usare */

/* connessione al server e selezione del database */
try {
  $pdo = new PDO("mysql:host=$dbhost;dbname=$dbname",
                 $dbuser, $dbpwd);
}
catch (PDOException $e) {
  print "Error!: " . $e->getMessage() . "<br />";
  die();
};
```


PDO: prepared statements

```
/* preparazione dello statement: media dei voti degli
   studenti di una provincia data */

$stmt = $pdo->prepare("
SELECT s.Matricola, s.Nome, s.Cognome, ROUND(AVG(e.Voto)) AS
Media
FROM Studenti s JOIN ESAMI e ON (s.Matricola=e.Candidato)
WHERE s.Provincia=:prov
GROUP BY s.Matricola, s.Nome, s.Cognome
");

/* associa la variabile $prov al parametro */
$stmt->bindParam(':prov', $prov);
```


PDO: prepared statements

```
/* Prima istanza: studenti di Venezia */
$prov="VE";

/* Esegue lo statement */
$stmt->execute();

/* Fetch dei risultati */

echo "<h3>Medie della  provincia di $prov</h3>";

while ($row=$stmt->fetch()) {
  $matricola=$row['Nome'];
  $nome=$row['Cognome'];
  $cognome=$row['Matricola'];
  $media=$row['Media'];
    echo "$matricola - $nome $cognome - $media<br />";
};
```


PDO: prepared statements

```
/* Seconda istanza: studenti di Padova */
$prov="PD";

/* Esegue lo statement */
$stmt->execute();

/* Fetch dei risultati */
...
```

Riassuntino

- Per il progetto:
  - leggere le informazioni fornite nella pagina del corso
  - programmazione PHP
  - interfaccia semplice!
    - pagine ad accesso libero
    - autenticazione
    - operazioni di ricerca/inserimento/cancellazione
  - dovete saper mostrare che l'avete fatto voi
  - niente librerie