WEP — WEb Programming

5 — PHP: query e sessioni

Lucidi per il corso di Basi di Dati tenuto da Paolo Baldan presso l'Università di Padova, anno accademico 2008/09

Mantenimento dello stato

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
- 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)

Mantenimento dello stato

- 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

Query String

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


```php
$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: Visualizzazione


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

...

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

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

QueryString1.php

Reload problem

- Se l'informazione di operazione avvenuta con successo (es. per un inserimento) è data dallo script che lo effettua, un reload causa un nuovo inserimento
- Con POST la risottomissione di dati "vecchi" viene segnalata, con GET no
- Soluzione: messaggio di successo in una pagina diversa

Hidden Field

Passaggio di parametri con hidden field

- 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

Esempio con hidden field

- **Esempio**: Wizard
  - 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 field

- 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 field

- 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 field

- 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

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'];
```

Cookie

Cookie

- 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, nel file system del client
  - rimandati al server ad ogni visita successiva del client

Cookie

- 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 cookie

- 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)
  - **path**: directory di validità (es: **"/" , "/Es03"**)
  - **domain**: dominio di validità (es. **".unipi.it"**)

- Da usare prima di qualsiasi output!

Accesso ai cookie

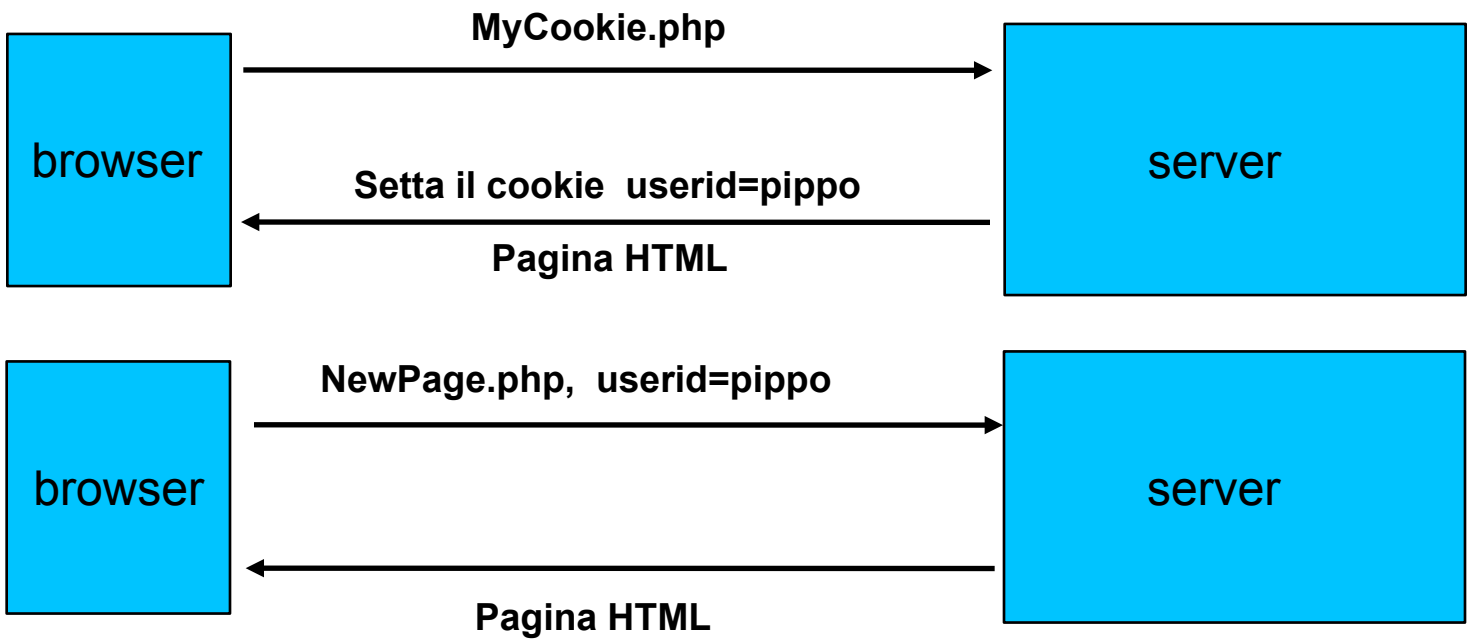
- Ai cookie si accede tramite il superglobal array **`$_COOKIE`**

  `$_COOKIE["name"]`

- Un cookie settato in una pagina è **visibile solo al caricamento successivo** (i cookie vengono inviati nello header secondo il protocollo HTTP)

Funzionamento dei cookie


```
<?PHP setcookie('userid','pippo'); ?>
<HTML><HEAD>...</HEAD>
<BODY>Pagina MyCookie.php che setta il cookie</BODY></HTML>
```



Cookie: 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 in ogni sottodir.) */
setcookie("visite", "$count", time()+60, "/");

/* l'output puo` cominciare */
...
if ($count)
  echo "Sei stato qui $count 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>";                                cookie-pers.php
...
echo "<A href=\"cookie-pers-form.php\">Cambia impostaz.</A>";
```

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>
...
```

Sessioni

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 o hidden field!!!)
  - 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**.

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 o è distrutta manualmente

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 distruggere 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[$sname])) {
    setcookie($sname,'', time()-3600,'/');
}
```

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 motore PHP 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 di inviare degli header HTTP **prima di ogni output** col comando

```
header(string)
```

- **Esempio**
  - Ridirezione ad altra pagina

```
<?PHP
header('Location: http://www.di.unipi.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 file.pdf */
readfile("original.pdf");
...
<!-- inizia il file HTML -->
<html>
  <body>
  ...
```

Esempio: session-form-check-0.php


```
require("util.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
- Interessante è lo script richiamato dall'ultima pagina, che distrugge la sessione

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

session_destroy();
$_SESSION=array();
$sname=session_name();
if (isset($_COOKIE[$sname])) {
    setcookie($sname,'', time()-3600,'/');
};
```

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, allora 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: Sessioni multiple


```
/* Alterna le due sessioni S1 e S2 nelle varie chiamate */
$name=$_GET["name"];
if (empty($name))
  $name="S1";

session_name("$name");
session_start();

if (empty($_SESSION['count'])) {
  $_SESSION['count'] = 1;
} else {
  $_SESSION['count']++;
}

if ($name=="S1")
  $nextname="S2";
else
  $nextname="S1";
```

SessioniMultiple.php

Esempio: Sessioni multiple


```
$count=$_SESSION['count'];
echo<<<END
  Sessione: $name
  Hai visitato questa pagina, nella sessione corrente,
  $count volte.
<P>
<a href="SessioniMultiple.php?name=$nextname">Continua</a>.
</P>
END;
```

SessioniMultiple.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 ripesca automaticamente

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-login-manage.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


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

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

session_destroy();

/* ed elimina il cookie corrispondente */
if (isset($_COOKIE['login'])) {
  setcookie($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!";
};
```

session-auth-register-manage.php

Autenticazione con cookie

- Stesse idee dell'autenticazione con sessioni
  - Quando l'utente effettua il login fornisce le proprie credenziali
  - Se passa il controllo viene 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 HTTP

- Il web-server può richieda 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"
```

  - 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 quindi accessibili come variabili di connessione

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

Autenticazione HTTP

- **Idea**
  - 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!");

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!");

echo "Bene! Eri autenticato!";
```

http-auth-operate1.php