PHP:
Sessioni e Connessione al DB

Prof. Viglietti Francesco
Www.in4matika.altervista.org

Le sessioni

Nella navigazione sui siti, ci sono delle situazioni in cui è importante stabilire un “dialogo” tra utente e server, in quanto ogni operazione che l'utente compie, può influenzare le successive ed essere influenzata dalle precedenti.

Pensiamo, ad esempio, ad un sistema di e-commerce, nel quale l'utente prima riempie un carrello scegliendo una serie di prodotti, poi viene portato sulla schermata che gli consente di effettuare il pagamento.

Il sistema deve "sapere" quali sono i prodotti che l'utente ha scelto e, di conseguenza, qual è la cifra che deve pagare. Siccome i prodotti sono stati scelti nelle schermate precedenti, il server ha bisogno di un sistema per ricollegare ciò che è stato fatto prima a ciò che deve fare ora.

Le sessioni

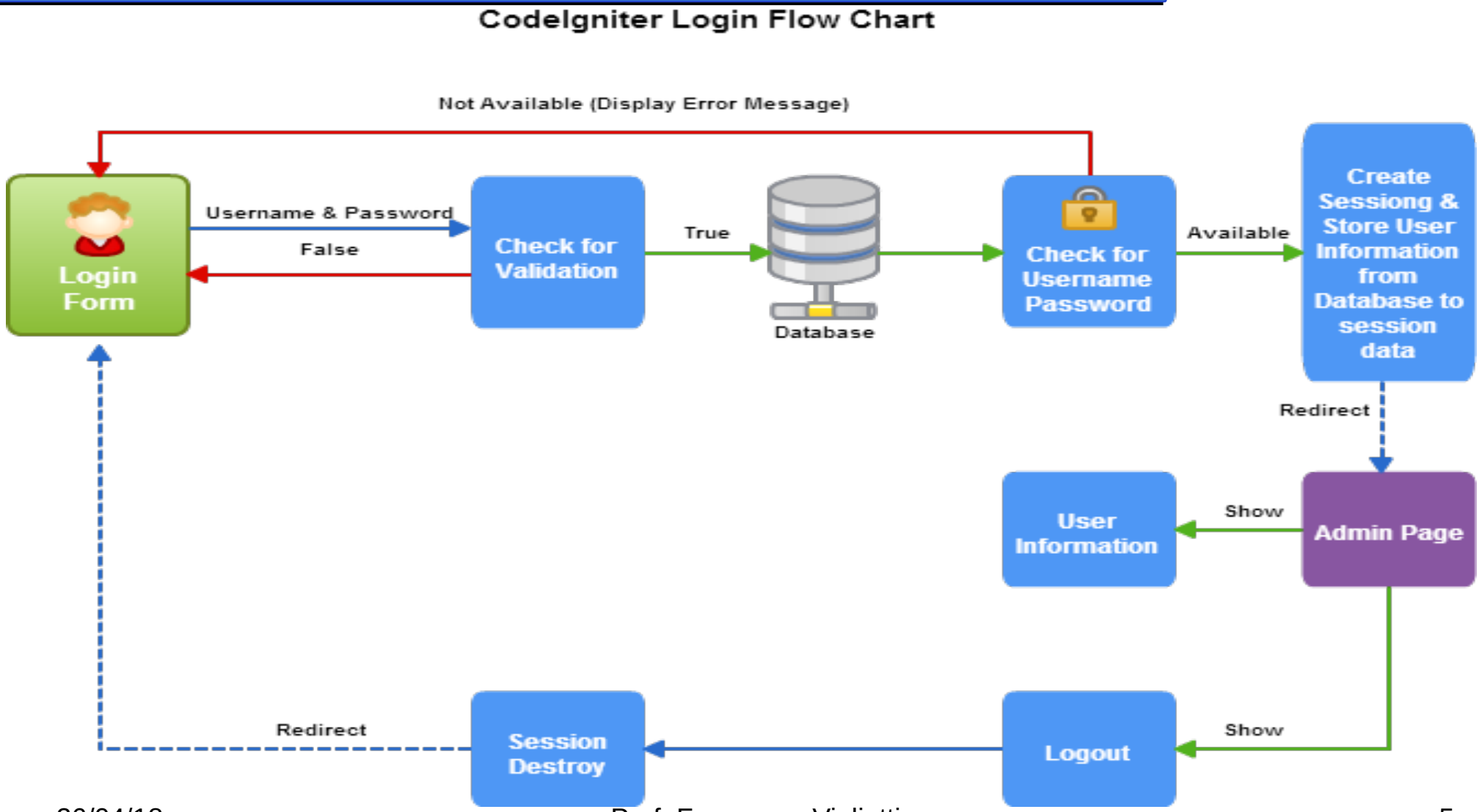
Un altro esempio, è quello dei siti con aree riservate: se un certo numero di pagine è visibile solo agli utenti registrati, questi dovranno effettuare un “riconoscimento” per poterle vedere, ma una volta effettuata l'***autenticazione*** (**login**) dovranno vedere tutte le pagine dell'area riservata, senza, ovviamente, doversi autenticare in ogni nuova pagina. Tutto questo sarà possibile solo se il server "***ricorda***" che quell'utente si è già logato.

Le sessioni

La prima cosa da fare quando si lavora con le sessioni è individuare dove salvare i files relativi. Le sessioni infatti sono files, che contengono i dati di un utente. Quindi si deve impostare nel file **php.ini**, la direttiva **session.save_path**, indicando la cartella dentro la quale verranno salvati i files. Di default, questa direttiva è impostata a "**/tmp**" che deve essere presente oppure si deve crearla. Altrimenti può essere indicato il percorso completo, es. "**C:\php\sessioni**".

L'importante è che questa cartella esista, altrimenti PHP non riuscirà a creare i files di sessione.

Le sessioni


CodeIgniter Login Flow Chart
Not Available (Display Error Message)
Login Form
Username & Password
False
Check for Validation
True
Database
Check for Username Password
Available
Create Sessiong & Store User Information from Database to session data
Redirect
Admin Page
Show
User Information
Show
Logout
Session Destroy
Redirect

Le sessioni

Quando si crea una sessione, PHP le assegna un nome, generando una sequenza casuale di lettere e numeri, quindi crea un file con quel nome nella cartella vista sopra, sarà quello il file di sessione identificativo dell'utente. Inoltre per riconoscere l'utente, PHP spedisce un cookie al suo browser, nel quale viene indicato il nome di questo file. In questo modo, quando l'utente farà una nuova chiamata al sito, il suo browser rispedirà il cookie con il nome della sessione, e PHP saprà da quale file recuperare i dati dell'utente. Tutto questo avviene solo fino a quando l'utente non chiuderà il browser, infatti a quel punto il cookie non sarà più valido, e la volta successiva l'utente dovrà ri-autenticarsi per essere riconosciuto.

Le sessioni

La funzione che permette di aprire una sessione è ***session_start()***, senza parametri.

Essa deve trovarsi in un punto dello script nel quale nessun output è ancora stato inviato al browser.

Successivamente PHP invierà il cookie al browser. Il cookie è un ***header***, cioè un tipo di dato che deve essere trasmesso prima del contenuto HTML della pagina.

Questo significa che nella parte precedente del file (rispetto alla chiamata di ***session_start()***) non deve esserci nulla all'esterno dei tag che delimitano il codice PHP.

Quando viene aperta la sessione, PHP crea un array superglobale di nome **$_SESSION**.

Tutti i dati dentro questo array vengono salvati sul file di sessione.

Le sessioni

Quindi, supponendo di voler salvare il nome e il cognome dell'utente che si è collegato:

session_start();
$_SESSION['nome'] = 'Giovanni';
$_SESSION['cognome'] = 'Bianchi';

I dati 'nome' e 'cognome' saranno salvati sul file di sessione, insieme ai valori relativi. In una chiamata successiva eseguita dallo stesso utente, dopo avere effettuato la ***session_start()*** avremo di nuovo a disposizione, nell'array ***$_SESSION***, gli stessi dati.

Le sessioni

Quando si esegue una session_start() possono aversi due possibilità:

1. Se il browser non ha spedito alcun cookie, la sessione deve essere creata, per cui PHP nomina la sessione, crea il file e spedisce il cookie
2. Se invece il browser ha spedito il cookie col nome della sessione, significa che la sessione è già aperta, e quindi PHP usa il nome della sessione per leggere il file relativo e rimettere a disposizione, nell'array **$_SESSION**, i dati che aveva salvato

Le sessioni

Per eliminare le variabili di sessione:

unset($_SESSION['nome']);
//elimina la variabile di sessione 'nome'

$_SESSION = array();
// elimina tutte le variabili di sessione

session_destroy(), elimina il file di sessione, ma il contenuto dell'array ***$_SESSION*** rimane disponibile per lo script in corso.

Quindi, se si vuole che il resto dello script tenga conto che la sessione non è più attiva, è necessario ricordarsi di svuotare l'array.

Il metodo GET e POST

La principale peculiarità del 'web dinamico', è la possibilità di variare i contenuti delle pagine in base alle richieste. Questa possibilità si materializza attraverso i meccanismi che permettono agli utenti, oltre che di richiedere una pagina ad un web server, anche di specificare determinati parametri che saranno utilizzati dallo script php per determinare quali contenuti la pagina dovrà mostrare. Come es., possiamo immaginare una pagina che visualizza le caratteristiche di un dato prodotto, prelevandole da un DB nel quale sono conservati i dati di un intero catalogo. Nel momento in cui si richiama la pagina, si dovrà specificare il codice del prodotto che deve essere visualizzato, per consentire allo script di prelevare dal DB i dati di quel prodotto e mostrarli all'utente.

Il metodo GET e POST

Vedremo come vengono salvati o recuperati i dati da un database, nel modo in cui PHP li riceve dall'utente. Esistono due sistemi per passare dati ad uno script: i metodi GET e POST.

Il metodo GET

Il metodo GET consiste nell'accodare i dati all'indirizzo della pagina richiesta, facendo seguire il nome della pagina da un ***?*** e dalle coppie nome/valore; le coppie sono separate da ***&*** e da =. Quindi, immaginando di avere la pagina ***prodotto.php*** che mostra le caratteristiche di un prodotto passandole il codice e la categoria del prodotto stesso, per visualizzare i dati del prodotto ***a7*** della categoria ***2***, dovremo richiamare la pagina in questo modo:

<a href="***prodotto.php?cod=a7&cat=2***">

La stringa che si trova dopo il ?, contenente nomi e valori dei parametri, viene detta ***querystring***. Quando la pagina prodotto.php viene richiamata in questo modo, essa avrà a disposizione, al suo interno, le variabili ***$_GET['cod']*** (con valore 'a7') e ***$_GET['cat']*** (con valore '2'). Infatti i valori contenuti nella query string sono memorizzati nell'array $_GET, che è un array superglobale in quanto è disponibile anche all'interno delle funzioni.

Il metodo POST

Il metodo di passaggio dei dati viene utilizzato da una pagina HTML che contiene il tag <form>, e l'attributo "method", può essere GET o POST. Se è GET, i dati vengono passati nella querystring. Se è POST, i dati vengono in maniera invisibile all'utente, con la richiesta HTTP che il browser invia al server. I dati passati con POST vengono memorizzati nell'array superglobale $_POST.

Quindi, per fare un esempio attraverso un piccolo modulo:

```
<form action="elabora.php" method="post">
<input type="text" name="nome">
<input type="checkbox" name="nuovo" value="si">
<input type="submit" name="submit" value="invia">
</form>
```

Questo modulo contiene una casella di testo chiamata 'nome' e una checkbox chiamata 'nuovo', il cui valore è definito come 'si'. Poi c'è il tasto di invio dati, con il metodo POST, alla pagina elabora.php.

Questa pagina troverà a disposizione la variabile $_POST['nome'], contenente il valore che l'utente ha digitato nel campo di testo; inoltre, se è stata selezionata la checkbox, riceverà la variabile $_POST['nuovo'] con valore 'si'. Attenzione però: se la checkbox non viene selezionata dall'utente, la variabile corrispondente risulterà non definita.

Utilizzare un database MySql

La possibilità di interagire con i DB è una delle potenzialità più interessanti offerte da PHP. I DB relazionali sono infatti lo strumento universalmente utilizzato per conservare basi di dati di qualsiasi dimensione, e PHP ci dà la possibilità di connetterci con un numero elevatissimo di DB server (MySql, Oracle, Ms Sql Server, Access, mSql ecc.).

Ci occuperemo dell'interazione con MySql, che è il DB server che si è affermato per la sua velocità e la sua stabilità, oltre che per il fatto di essere open source. Ci occuperemo delle tecniche da utilizzare per stabilire una connessione a MySql da uno script PHP, in modo da poter leggere e manipolare i dati. L'interazione viene ottenuta attraverso una serie di funzioni, il cui nome inizia sempre per "mysqli_" e dall'indicazione specifica della funzione.

Connessione al database MySql

La prima cosa da fare, è stabilire il collegamento con il server MySql. A tale scopo si utilizza la funzione

mysqli_connect(server, utente, password, database),

alla quale passiamo l'indirizzo su cui si trova il server MySql ('localhost' se gira sulla stessa macchina del web server), il nome utente di collegamento, la sua password e il nome del database in MySql. Questa funzione, se il collegamento è riuscito, restituisce un identificativo della connessione, cioè una variabile di tipo resource, che serve come "puntatore" al DB.

Se il collegamento non riesce, la funzione restituisce il valore booleano FALSE.

Connessione al database MySql

Si verifica che valore ha restituito la connessione. Per fare questo, generalmente si usa una sintassi di questo tipo:

$conn = mysqli_connect('localhost','mario','xxx','Dbname') or die("Errore nella connessione a MySql: " . mysqli_error());

Questa istruzione cerca di connettersi al Dbname del server MySql in localhost, con l'utente 'mario' e la password 'xxx'. La or si esegue solo nel caso mysqli_connect() restituisca FALSE, (connessione non riuscita), e quindi viene eseguita la funzione die(), che termina l'esecuzione dello script stampando ciò che è indicato fra parentesi e la funzione mysqli_error() che stampa l'errore segnalato dal server MySql. Se invece è riuscita, la variabile $conn contiene il nostro identificativo di connessione.

Eseguire una query

Per eseguire la query si usa la funzione mysqli_query(connessione,query), alla quale viene passata la query da eseguire insieme all'identificativo di connessione già visto prima. Anche questa funzione restituisce un valore, per il quale però dobbiamo distinguere due possibilità rispetto al tipo di query che abbiamo lanciato:

Se si tratta di una query di interrogazione (SELECT, SHOW, EXPLAIN, DESCRIBE), la funzione restituisce una variabile di tipo resource, che servirà successivamente, se la query è andata a buon fine; se invece MySql ha rilevato degli errori, la funzione restituisce FALSE;

Eseguire una query

Se invece si tratta di una query di aggiornamento (INSERT, UPDATE, DELETE,...), la funzione restituirà in ogni caso un valore booleano, ad indicare se l'esecuzione è andata a buon fine.

Vediamo quindi un esempio:

- ***$query = 'SELECT * FROM tabella';***
- ***$ris = mysqli_query($conn,$query) or die("Errore nella query: ". mysqli_error());***

Se la query ha avuto successo **$ris** conterrà l'identificativo del risultato, che ci servirà successivamente per leggere le righe restituite dal db. Se invece la query non va a buon fine, lo script si blocca segnalando l'errore.

Verifica dei risultati della query

Può verificarsi che una query, pur essendo corretta, non produce alcun risultato, ad es. perché le condizioni che abbiamo specificato nella clausola WHERE non sono mai verificate sulle tabelle interessate.

Se vogliamo sapere quante righe sono state restituite da una SELECT, possiamo usare la funzione **mysqli_num_rows(risultato)**, che restituisce il numero di righe contenute dall'identificativo del risultato che le passiamo.

Se invece abbiamo eseguito una query di aggiornamento e vogliamo sapere quante righe sono state modificate, possiamo usare **mysqli_affected_rows(connessione)**, che restituisce il numero di righe modificate dall'ultima query di aggiornamento.

Verifica dei risultati della query

```
$query = 'SELECT * FROM tabella';
$ris = mysqli_query($conn,$query) or die("Errore nella query: " . mysqli_error());
$righe = mysqli_num_rows($ris);
/* $righe riceve il num. di righe restituite dalla select */
$query="UPDATE tabella SET campo1='valore' WHERE campo2='val'";
mysqli_query($conn,$query) or die("Errore nella query: " . mysqli_error());
$righe = mysqli_affected_rows($conn);
/* $righe riceve il num. di righe modificate dall'UPDATE */
```

E' importante notare la differenza nel parametro da passare alle due funzioni: ***mentre mysqli_num_rows()*** richiede un identificativo di risultato, ***mysqli_affected_rows()*** richiede un identificativo di connessione.

Lettura dei risultati di una SELECT

Una volta effettuata una query di interrogazione, per leggerla si utilizza la funzione ***mysqli_fetch_array(risultato)***, la quale, ogni volta che viene chiamata, restituisce una riga del risultato; quando non ci sono più righe da leggere, restituisce FALSE. Quindi, per scorrere tutto il risultato usiamo questa funzione come condizione di un ciclo, che si concluderà quando restituisce FALSE.

In questo modo evitiamo di sapere a priori quante sono le righe contenute nel risultato stesso. Questa variabile è un array che contiene i valori delle colonne restituiti dalla query. Gli indici dell'array sono i nomi delle colonne, ed i loro valori sono i valori estratti dal DB.

Lettura dei risultati di una SELECT Esempio:

```
$query = 'SELECT nome, indirizzo, telefono FROM tabella';
    $ris = mysqli_query($conn,$query) or die("Errore nella query: " .
    mysql_error());
    while($riga = mysqli_fetch_array($ris)){
        print"Nome: $riga[nome]<br>"
        print"Indirizzo: $riga[indirizzo]<br>"
        print"Telefono: $riga[telefono]<br><hr>"
}
```

Con questo ciclo si stamperanno tutti i valori estratti dalla query, separando con una riga vuota i blocchi relativi ad ogni record. Dalla Select verranno estratte le colonne 'nome', 'indirizzo' e 'telefono', e quindi l'array $riga conterrà tre elementi con questi indici.

La funzione ***mysqli_close(connessione)***, chiude la connessione aperta con ***mysqli_connect()***, anche se PHP chiude automaticamente connessioni aperte.