

Autenticazione e autorizzazione per applicazioni Web

...

Concetti introduttivi

Autenticazione

- Processo con cui si verifica il possesso di credenziali di accesso a un servizio, non che qualcuno sia chi dice di essere.
- Esempi:
 - login con nome utente e password (la più semplice, quella che useremo noi)
 - autenticazione con 2 fattori
 - utilizzo di parametri biometrici
- Se si vogliono associare le credenziali con una persona (o altro) c'è bisogno di un riconoscimento preventivo

Autorizzazione

- Generalmente è il passo successivo all'autenticazione, una volta riconosciuto l'utente vengono associati dei permessi su cosa può fare o non può fare
- Le autorizzazioni sono sia a livello di classe di utenza, sia al livello del singolo utente
- **Esempio per classe di utenza:** tutti i docenti possono inserire un voto nel registro elettronico, nessuno studente lo può fare
- **Esempio per singolo utente:** ogni docente può inserire voti nella propria classe/disciplina, nessuno può farlo per classi/discipline di altri docenti

Autenticazione in PHP per applicazioni Web

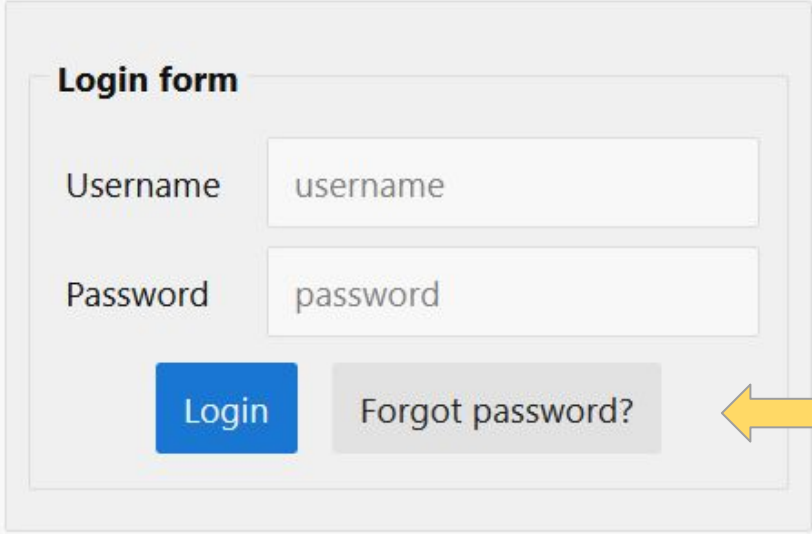
- Il modello che seguiremo è quello già visto per le applicazioni precedenti, in cui gli script PHP gestiscono l'input dell'utente, si interfacciano con il database e rispondono con pagine HTML
- Ci sono due ordini di problemi:
 - gestire l'associazione nome utente - password
 - evitare che l'utente non debba reinserirla tutte le volte che si sposta da una pagina ad altre sezioni del sito (ricordarsi che il protocollo *http* è stateless)

Verifica delle credenziali

- Il modo più semplice e anche il più comune per verificare le credenziali è quello di far comunicare all'utente una coppia username-password, che verranno controllate ogni volta che l'utente vorrà accedere all'applicazione
- La password viene memorizzata nel database sotto forma di hash con un algoritmo crittografico adeguato, perchè?
- In PHP esiste la funzione **password_hash** che permette di creare una password con un algoritmo adeguato, per verificarla si usa **password_verify**.

Come acquisire le credenziali

- Basta un semplice form HTML



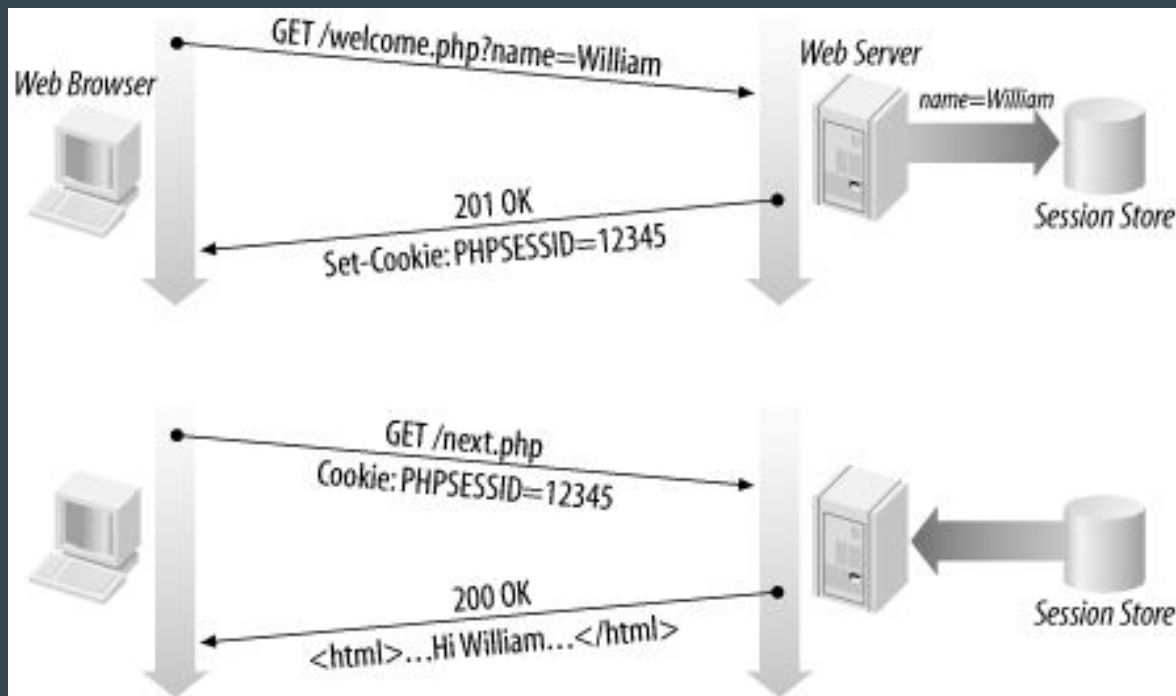
The image shows a login form titled "Login form". It contains two input fields: "Username" with the placeholder text "username" and "Password" with the placeholder text "password". Below the fields are two buttons: a blue "Login" button and a grey "Forgot password?" button. A yellow arrow points from the "Forgot password?" button towards the right, indicating the next step in the process.

Per recuperare la password l'unico modo possibile è crearne una nuova, solitamente si passa attraverso l'email

E dopo l'autenticazione?

- Una volta che siano state riconosciute le credenziali, come è possibile che nella navigazione in altre pagine lo stesso utente venga riconosciuto, senza che gli vengano nuovamente richieste le credenziali, essendo *http* stateless?
- In PHP esiste il meccanismo delle *sessioni* che scarica il programmatore dall'inventarsi tecniche per fare in modo che l'applicazione si ricordi informazioni nel passaggio da una richiesta a un'altra.
- Per far questo viene utilizzato un *cookie* gestito in automatico, che è una chiave univoca che lato client viene memorizzata e rimandata al server a ogni accesso, e il server usa per tenere associate tutte le informazioni che il programmatore ritiene utili

Meccanismo delle sessioni in PHP



Cosa usare con PHP

- Ogni pagina sottoposta a autenticazione deve far partire la sessione con l'istruzione `session_start()` e poi fare i controlli opportuni
- All'interno della pagina vengono rese disponibili tutte le informazioni di sessione attraverso il super array globale `$_SESSION`
- Le informazioni dentro la variabile `$_SESSION` sono memorizzate sul server, quindi non sono modificabili dal client
- Per fare il logout viene distrutta la sessione sia lato client (eliminando il cookie), sia lato server (eliminando il file che contiene le informazioni associate)

Esempio (reperibile su GitHub)



E per l'autorizzazione?

- L'idea di base è semplice, basta associare ai vari utenti delle classi diverse, ad esempio studente, docente, dirigente, ecc. e ad ognuna di queste classi le operazioni che possono svolgere
- Sia le classi che i permessi possono essere memorizzati nel database, i permessi potrebbero essere associazioni del tipo classe-operazioni, ad esempio studente-visualizza_voti, docente-inserisci_voti, ecc.
- Un'implementazione scalabile sicuramente non è banale, esistono librerie che permettono di risolvere questi problemi, non verrà comunque approfondita questa parte.