

Cookie in PHP

Gestione della Persistenza nelle Applicazioni Web

Prof. Fedeli Massimo IIS E. Fermi Sacconi Cpia

Ascoli Piceno

5 gennaio 2026

- 1 Il Protocollo HTTP
- 2 Formato dei Messaggi HTTP
- 3 Header HTTP e Cookie
- 4 Persistenza in PHP
- 5 I Cookie
- 6 Normativa sui Cookie

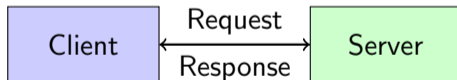
Il Protocollo HTTP

Hyper Text Transfer Protocol

Il World Wide Web per il trasferimento di dati ipertestuali si basa sul protocollo applicativo **HTTP** che utilizza l'architettura client/server.

Caratteristiche principali:

- Protocollo di livello applicativo
- Basato su TCP/IP
- Scambio di messaggi testuali
- RFC 1945 (HTTP/1.0)
- RFC 2616 (HTTP/1.1)



Struttura dei Messaggi HTTP

Tipologie di Messaggi

I messaggi HTTP sono composti da caratteri ASCII quindi leggibili e terminati da CR+LF (Carriage Return + Line Feed)

- Una riga vuota (ovvero due CR+LF di fila) indica che l'intestazione è finita e che sta per iniziare il "**corpo**" del messaggio (ad esempio il codice HTML della pagina o i dati di un modulo).
- Un esempio pratico: Ecco come appare una richiesta per il sito esempio.it:
GET /index.html HTTP/1.1[CRLF] Host: esempio.it[CRLF] Accept: text/html[CRLF]
[CRLF] j- Riga vuota (fine intestazioni)
- Senza quei segnali CR+LF, il server vedrebbe tutto come un unico blocco di testo confuso (GET /index.html HTTP/1.1Host: esempio.it...) e restituirebbe un errore.

Struttura dei Messaggi HTTP

Tipologie di Messaggi

I messaggi HTTP possono essere messaggi di richiesta o di risposta:

REQUEST (Richiesta)

- Inviata dal client
- Richiede una risorsa
- Contiene metodi (GET, POST, ecc.)

RESPONSE (Risposta)

- Inviata dal server
- Fornisce la risorsa richiesta
- Contiene codici di stato

Importante

La comunicazione avviene mediante TCP/IP utilizzando gli indirizzi IP dei computer che ospitano client e server.

Meccanismo di Comunicazione HTTP

Sequenza di Operazioni

1. Apertura connessione TCP



2. Browser richiede risorsa



3. Server invia risposta



4. Chiusura connessione

Nota

Ogni richiesta di pagina richiede una nuova connessione TCP indipendente.

Esempio di Connessione HTTP

Fasi della Connessione

- 1 **Analisi URL:** Il browser estrae il dominio dall'URL

```
http://www.esempio.it/pagina.html
```

- 2 **Connessione TCP:** Il client inizia una connessione verso il server sulla porta 80
- 3 **Invio Richiesta:** Il client invia una richiesta GET attraverso il socket TCP

```
GET /pagina.html HTTP/1.1  
Host: www.esempio.it
```

- 4 **Risposta Server:** Il server incapsula la risorsa nella risposta HTTP e la invia al client.

Completamento della Connessione HTTP

Fasi Finali

- 5 **Chiusura Connessione:** Il server richiede al TCP di chiudere la connessione dopo l'invio della risposta
- 6 **Terminazione TCP:** La connessione si conclude dopo il riscontro del client
- 7 **Parsing HTML:** Il client estrae il file HTML e identifica gli oggetti referenziati (immagini, CSS, JavaScript)
- 8 **Richieste Multiple:** I passi precedenti vengono ripetuti per ogni risorsa referenziata (eventualmente aprendo connessioni in parallelo)

Prestazioni

L'apertura di multiple connessioni parallele migliora le prestazioni nel caricamento delle pagine web moderne.

Formato dei Messaggi HTTP

Componenti Principali

Struttura:

① **START-LINE**

Riga di richiesta/risposta

② **HEADER**

Intestazione HTTP

③ **BODY**

Corpo HTTP (opzionale)

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html
[riga vuota]
[eventuale body]
```

Metodi HTTP Principali

GET, POST, HEAD, PUT, DELETE, TRACE, CONNECT, OPTIONS

START-LINE - Riga di Richiesta

Elementi della Prima Riga

La START-LINE della richiesta contiene tre elementi:

METODO

GET, POST, ...

PERCORSO

/index.html

VERSIONE

HTTP/1.1

Esempio

```
GET /sistemi/index.html HTTP/1.1  
POST /form/login.php HTTP/1.1
```

Formato dei Messaggi HTTP

X Headers Payload Preview Response Initiator Timing Cookies	
▼ General	
Request URL	https://www[REDACTED].it/gymcloud/api/abbonamenti_list.php?tipo=I
Request Method	GET
Status Code	200 OK
Remote Address	89.46.110.51:443
Referrer Policy	strict-origin-when-cross-origin
▼ Response Headers	
Alt-Svc	h3=":443"; ma=86400
Cache-Control	no-store, no-cache, must-revalidate
Content-Encoding	gzip
Content-Type	application/json
Date	Mon, 05 Jan 2026 09:03:38 GMT
Expires	Thu, 19 Nov 1981 08:52:00 GMT

Nella sezione General, visibile tramite i webtools del browser troviamo tra le altre informazioni

Header HTTP - Intestazione

Campi dell'Header

L'intestazione contiene informazioni aggiuntive sulla richiesta/risposta

Informazioni nell'Header:

- Tipo di browser
- Data e ora
- Cookie
- Codifica
- Lingua preferita
- Tipo di connessione

```
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html
Accept-Language: it-IT
Accept-Encoding: gzip
Connection: keep-alive
Cookie: session=abc123
```

Body del Messaggio

Il corpo può essere omesso nella richiesta, mentre è sempre presente nella risposta con la pagina HTML.

Esempio Completo - Richiesta HTTP

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Accept: text/html,application/xhtml+xml
Accept-Language: it-IT,it;q=0.9,en;q=0.8
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cache-Control: max-age=0
Cookie: sessionId=xyz789; user=mario
```

- **Linea 1:** Metodo GET, risorsa richiesta, versione HTTP
- **Linee 2-9:** Header con metadati della richiesta
- **Linea 10:** Riga vuota che separa header da body
- **Body:** Assente nelle richieste GET

Esempio Completo - Risposta HTTP

```
HTTP/1.1 200 OK
Date: Mon, 15 Jan 2024 10:30:00 GMT
Server: Apache/2.4.41
Content-Type: text/html; charset=UTF-8
Content-Length: 1234
Connection: keep-alive
Set-Cookie: sessionId=abc123; Path=/; HttpOnly

<!DOCTYPE html>
<html>
<head><title>Pagina di Esempio</title></head>
<body><h1>Benvenuto!</h1></body>
</html>
```

- **Linea 1:** Versione HTTP e codice di stato
- **Linee 2-7:** Header con metadati della risposta
- **Linee 9-13:** Body con il contenuto HTML

Metodo POST - Invio Dati

Utilizzo del Body

Il corpo del messaggio contiene i dati di un form **SOLO** con il metodo POST

```
POST /login.php HTTP/1.1
Host: www.example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 35

username=mario&password=secretpass
```

GET vs POST:

- GET: parametri nell'URL
- POST: parametri nel body
- POST: più sicuro e capiente

Sicurezza

Il metodo POST è preferibile per dati sensibili (password, dati personali)

Significato dei campi dell'Header - 1. Identificazione e Controllo

Gli header di identificazione definiscono l'origine e la destinazione della richiesta.

- **Host:** Dominio del server (es. `google.com`). Obbligatorio in HTTP/1.1.
- **User-Agent:** Identifica browser e sistema operativo del client.
- **Referer:** URL della pagina precedente che ha generato il link.
- **Origin:** Indica la provenienza di una richiesta (fondamentale per la sicurezza CORS).

Significato dei campi dell'Header - 2. Negoziazione del Contenuto

Permettono al client di comunicare al server i formati preferiti per la risposta.

Campi "Accept"

- **Accept:** Tipi MIME supportati (es. `text/html`, `application/json`).
- **Accept-Encoding:** Algoritmi di compressione (es. `gzip`, `br`).
- **Accept-Language:** Lingue preferite (es. `it-IT`, `en-US`).
- **Accept-Charset:** Set di caratteri (es. `utf-8`).

Significato dei campi dell'Header - 3. Sicurezza e Gestione Sessione

Gestiscono l'accesso alle risorse protette e il mantenimento dello stato.

Authorization Credenziali per l'accesso (es. Bearer token o Basic auth).

Cookie Invia al server i token di sessione precedentemente memorizzati.

Upgrade-Insecure-Requests Segnala al server la preferenza per connessioni cifrate (HTTPS).

DNT (Do Not Track) Esprime la volontà dell'utente di non essere tracciato.

Significato dei campi dell'Header - 4. Header Condizionali e Caching

Fondamentali per ridurre il traffico dati e migliorare le prestazioni.

- **Cache-Control:** Direttive per la cache (es. `no-cache`, `max-age`).
- **If-None-Match / If-Modified-Since:** La risorsa viene inviata solo se è cambiata rispetto alla versione in cache (basato su ETag o data).
- **Connection:** Gestione della persistenza TCP (es. `keep-alive`).

5. Dati dell'Entità e Client Hints

Usati quando il client invia dati (es. POST) e per la privacy moderna.

- **Content-Type:** Formato dei dati inviati (es. `multipart/form-data`).
- **Content-Length:** Dimensione in byte del corpo della richiesta.
- **Sec-CH-UA:** (Moderno) Versione sicura dello User-Agent per proteggere la privacy.

Nota: Tutti gli header seguono la struttura `Chiave: Valore`.

Codici di Stato HTTP

Categorie dei Codici di Stato

Il server risponde con un codice numerico che indica l'esito della richiesta

Codice	Significato
1xx	Informational - Richiesta ricevuta
2xx	Success - Richiesta completata con successo 200 OK - Richiesta riuscita
3xx	Redirection - Ulteriori azioni necessarie 301 Moved Permanently - Risorsa spostata
4xx	Client Error - Errore nella richiesta 404 Not Found - Risorsa non trovata
5xx	Server Error - Errore del server 500 Internal Server Error

Elementi dell'Header HTTP

Meta-informazioni

Ogni volta che un utente visita una pagina web, browser e server si scambiano meta-informazioni mediante l'header HTTP

Campi dell'Header:

- Ogni riga è chiamata "**Campo dell'Header**"
- Formato: Nome: Valore
- Separatore: due punti (:)
- Circa 100 campi disponibili
- 30 campi per la richiesta
- 30 campi per la risposta
- Altri campi non standardizzati

Principali Campi dell'Header

Significato dei Campi Comuni

Campo	Descrizione
HTTP/1.1	Versione del protocollo HTTP utilizzata
200 OK	Codice di stato (ricezione e accettazione)
Content-Encoding	Tipo di codifica del file
Content-Type	Tipo MIME del contenuto
Age, Cache-Control	Informazioni sul caching
Expires, Vary	Gestione della cache
ETag	Versione del file (per validazione cache)
Last-Modified	Data ultima modifica
Server	Software del web server
Content-Length	Dimensione del file in byte

Scopo dell'Header HTTP

Funzioni Principali

Le informazioni dell'header servono per il coordinamento tra client e server

Obiettivi dello Scambio di Header:

- **Comprensione formato:** Assicurare che il client comprenda la forma del file ricevuto
- **Validazione dimensione:** Verificare che la dimensione coincida con quella attesa
- **Gestione cache:** Ottimizzare le prestazioni memorizzando risorse
- **Negoziazione contenuto:** Selezionare la versione più adatta (lingua, formato)
- **Gestione sessioni:** Mantenere lo stato attraverso cookie

Estensibilità

Esistono quasi 100 campi header, di cui solo una parte è standardizzata

Campo Set-Cookie - Risposta del Server

Creazione di Cookie

Il campo Set-Cookie nella risposta del server richiede la memorizzazione di informazioni in un cookie sul client

```
HTTP/1.1 200 OK
Date: Mon, 15 Jan 2024 10:30:00 GMT
Server: Apache/2.4.41
Content-Type: text/html; charset=UTF-8
Set-Cookie: username=mario; Expires=Wed, 15-Jan-2025 10:30:00 GMT
Set-Cookie: sessionId=xyz789; Path=/; HttpOnly; Secure

<!DOCTYPE html>
...
```

Importante

Il server può richiedere la creazione di multipli cookie nella stessa risposta

Campo Cookie - Richiesta del Client

Invio Cookie al Server

Il campo Cookie nella richiesta del client comunica il contenuto archiviato nei cookie

```
GET /profilo.php HTTP/1.1  
Host: www.example.com  
User-Agent: Mozilla/5.0  
Cookie: username=mario; sessionId=xyz789  
Accept: text/html
```



La Persistenza in PHP

Protocollo Stateless

Il protocollo HTTP non permette al server di "riconoscere" un utente che si era precedentemente collegato

Caratteristiche di HTTP:

- Ogni pagina richiede una connessione TCP **indipendente**
- Nessuna informazione viene mantenuta tra richieste successive
- Ogni coppia request/response è isolata dalle altre
- **HTTP è un protocollo STATELESS**

Conseguenza

Le richieste dei client non lasciano alcuno stato nel server. Ciascuna coppia request_client/response_server è indipendente dalle altre.

Limiti del Protocollo Stateless

Problemi Applicativi

Sin dall'inizio dell'espansione del Web, questa caratteristica del protocollo HTTP ha mostrato tutti i suoi limiti

Scenari Problematici:

- **Autenticazione:** Come mantenere l'utente loggato?
- **Carrello e-commerce:** Come ricordare i prodotti selezionati?
- **Preferenze:** Come salvare le impostazioni dell'utente?
- **Tracking:** Come seguire il percorso di navigazione?

Soluzione

Esistono tecniche che simulano lo stato in una tipica connessione client/server

Tecniche per la Persistenza della Connessione

Il Problema

Dopo che una pagina web viene inviata dal server al client, per uno script PHP non è più possibile accedere ai dati relativi alla pagina stessa

Necessità degli Sviluppatori:

- Memorizzare informazioni persistenti per più pagine
- Mantenere traccia degli utenti loggati
- Conservare preferenze e impostazioni

Soluzioni in PHP

PHP mette a disposizione due metodi principali:

- 1 **COOKIE** - Memorizzazione lato client
- 2 **SESSIONI** - Memorizzazione lato server

Definizione

I cookie sono **file di testo** memorizzati sul client su richiesta esplicita del server

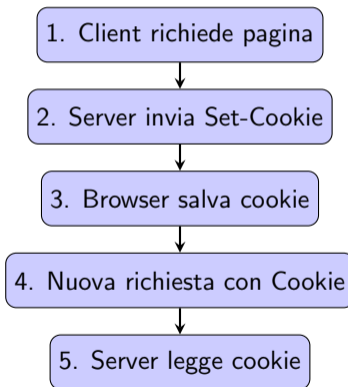
Caratteristiche:

- File di piccole dimensioni
- Memorizzati sul filesystem del client
- Associati a un dominio specifico
- Validità temporale configurabile
- Inviati automaticamente nelle richieste

Funzionamento:

- 1 Server richiede creazione via Set-Cookie
- 2 Browser salva il cookie
- 3 Nelle richieste successive il cookie viene inviato
- 4 Server legge i dati del cookie

Processo di Gestione



Componenti di un Cookie

Elementi Fondamentali

I cookie sono caratterizzati da diversi parametri

Elemento	Descrizione
Nome	Identificatore univoco del cookie
Valore	Dati memorizzati nel cookie
Scadenza	Validità temporale (opzionale)
Path	Percorso del sito per cui è valido
Domain	Dominio per cui è valido
Secure	Trasmissione solo su HTTPS
HttpOnly	Non accessibile da JavaScript

Nota

Impostazioni di sicurezza adeguate proteggono i dati degli utenti

Array Superglobale \$_COOKIE

Accesso ai Cookie in PHP

\$_COOKIE è un array associativo che memorizza i valori dei cookie inviati dal client

```
<?php
// Lettura di un cookie
if (isset($_COOKIE['username'])) {
    $username = $_COOKIE['username'];
    echo "Benvenuto, " . $username;
} else {
    echo "Cookie non trovato";
}

// Visualizzare tutti i cookie
foreach ($_COOKIE as $nome => $valore) {
    echo "$nome: $valore<br>";
}
?>
```

Creazione di Cookie - Esempio Base

Funzione setcookie()

```
<?php
$dato = "Questa stringa viene memorizzata nel cookie";

// Cookie senza scadenza (session cookie)
setcookie("Alfa", $dato);

// Cookie con scadenza di 30 giorni
setcookie("Beta", $dato, time() + 60*60*24*30);
?>
```

Cookie "Alfa":

- Senza scadenza
- Valido per la sessione

Cookie "Beta":

- Scadenza: 30 giorni
- Persistente

Calcolo della Scadenza del Cookie

Funzione time()

La funzione `time()` restituisce il timestamp Unix corrente (secondi dal 1 gennaio 1970)

```
<?php
// Cookie valido per 30 giorni
setcookie("Beta", $dato, time() + 60*60*24*30);

// Calcolo dettagliato:
// 60 secondi = 1 minuto
// 60 minuti = 1 ora (60 * 60)
// 24 ore = 1 giorno (60 * 60 * 24)
// 30 giorni = 1 mese (60 * 60 * 24 * 30)

// Altri esempi di scadenza:
setcookie("temp", "valore", time() + 3600); // 1 ora
setcookie("week", "valore", time() + 604800); // 1 settimana
setcookie("year", "valore", time() + 31536000); // 1 anno
```

Funzione setcookie() - Sintassi Completa

Prototipo Completo

```
setcookie(  
    string $name,  
    string $value = "",  
    int $expires = 0,  
    string $path = "",  
    string $domain = "",  
    bool $secure = false,  
    bool $httponly = false  
): bool
```

Parametro	Descrizione
name	Nome del cookie (obbligatorio)
value	Valore da memorizzare
expires	Timestamp di scadenza (0 = session cookie)
path	Percorso sul server (default: directory corrente)
domain	Dominio per cui è valido
secure	TRUE = solo HTTPS
httponly	TRUE = non accessibile da JavaScript

Restrizioni della Funzione setcookie()

IMPORTANTE

I cookie devono essere inviati **prima** di qualsiasi output dello script!

ERRATO:

```
<?php
echo "Pagina Web";
// ERRORE: header gi inviati!
setcookie("test", "valore");
?>
```

CORRETTO:

```
<?php
// Prima setcookie
setcookie("test", "valore");
// Poi output
echo "Pagina Web";
?>
```

Spiegazione

Questa è una restrizione del protocollo HTTP: l'header (che contiene Set-Cookie) deve precedere il body

Soluzione

Usare `ob_start()` per il buffering dell'output se necessario

Esempio Completo - Gestione Cookie

```
<?php
// Imposta un cookie con tutte le opzioni
setcookie(
    "user_prefs",           // nome
    "theme=dark&lang=it",   // valore
    time() + 86400 * 30,    // scadenza: 30 giorni
    "/",                   // path: tutto il sito
    "example.com",          // domain
    true,                   // secure: solo HTTPS
    true                    // httponly: protetto da XSS
);

// Lettura del cookie
if (isset($_COOKIE['user_prefs'])) {
    parse_str($_COOKIE['user_prefs'], $preferences);
    echo "Tema: " . $preferences['theme'];
    echo "Lingua: " . $preferences['lang'];
}
?>
```

Eliminazione di un Cookie

Metodo

Un cookie può essere eliminato impostando una scadenza **precedente** all'orario corrente

```
<?php
// Eliminazione di un cookie
setcookie("nome_cookie", "", time() - 3600);

// Oppure con data esplicita nel passato
setcookie("nome_cookie", "", 1);

// Importante: usare gli stessi parametri di path e domain
// utilizzati in fase di creazione
setcookie("user_prefs", "", time() - 3600, "/", "example.com");

// Verifica eliminazione
if (!isset($_COOKIE['nome_cookie'])) {
    echo "Cookie eliminato con successo";
}
```

Cookie - Vantaggi e Svantaggi

Vantaggi

- Semplici da implementare
- Non caricano il server
- Persistenti nel tempo
- Supportati da tutti i browser
- Permettono personalizzazione

Svantaggi

- Memorizzati lato client
- **Modificabili dall'utente**
- Limite di dimensione (4KB)
- Privacy concerns
- Possono essere disabilitati
- Non sicuri per dati sensibili

Punto Debole Principale

Le informazioni di stato vengono salvate in un file sul filesystem del client e sono quindi **potenzialmente modificabili** dall'utente

Riferimenti Normativi

- **Provvedimento del Garante** per la Protezione dei Dati Personali (Gazzetta Ufficiale n. 126 del 3 giugno 2014)
- **Linee guida cookie e altri strumenti di tracciamento** – 10 giugno 2021 [9677876] (Gazzetta Ufficiale n. 163 del 9 luglio 2021)

Classificazione

La normativa distingue tra:

- 1 **Cookie Tecnici**
- 2 **Cookie di Profilazione**

Definizione

I cookie tecnici sono quelli che facilitano la navigazione permettendo al sito di offrire alcune funzionalità

Esempi di Cookie Tecnici:

- **Autenticazione:** Riconoscimento dell'utente loggato senza dover reinserire username e password
- **Carrello:** Mantenimento dei prodotti aggiunti anche dopo giorni
- **Preferenze lingua:** Memorizzazione della lingua scelta
- **Impostazioni visualizzazione:** Font size, tema, layout

Caratteristica Importante

I cookie tecnici **non richiedono** il consenso esplicito dell'utente

Tre Tipologie

1 Cookie di navigazione o di sessione

- Garantiscono la normale navigazione e fruizione del sito web
- Permettono acquisti e autenticazione ad aree riservate
- Durata limitata alla sessione

2 Cookie analytics

- Assimilati ai cookie tecnici se usati dal gestore del sito
- Raccolgono informazioni aggregate sul numero di utenti
- Analizzano come gli utenti visitano il sito

3 Cookie di funzionalità

- Permettono navigazione personalizzata
- Memorizzano criteri selezionati (lingua, prodotti)
- Migliorano il servizio offerto all'utente

Definizione Normativa

”I Cookie di profilazione hanno come scopo la creazione di un profilo del navigatore”

Finalità:

- Comprendere il comportamento sul sito
- Identificare interessi e orientamenti
- Raccogliere e incrociare informazioni (proprie o di terze parti)
- Capire chi è l'utente e cosa gli interessa

Utilizzo:

- Invio di messaggi pubblicitari personalizzati
- Vendita di servizi o prodotti mirati
- Personalizzazione navigazione oltre il minimo necessario

IMPORTANTE

Obbligo di Informativa

I siti web devono informare gli utenti sull'uso dei cookie e richiedere il consenso

Requisiti del Banner:

- **Chiario e visibile** al primo accesso
- **Informazioni complete** sui cookie utilizzati
- **Scelta granulare** per diverse tipologie di cookie
- **Possibilità di rifiuto** facilmente accessibile
- Link alla **Cookie Policy** completa

Best Practice

- Cookie tecnici: attivati automaticamente
- Cookie di profilazione: richiedono consenso esplicito
- Permettere personalizzazione delle scelte

Conclusioni e Alternative ai Cookie

Recap Cookie

I cookie rappresentano una soluzione semplice ma con limitazioni di sicurezza

Quando Usare i Cookie:

- Dati non sensibili
- Preferenze utente
- Tracking analytics
- Personalizzazione base

Alternative più Sicure

- **SESSIONI PHP:** Dati memorizzati lato server
- **Database:** Persistenza a lungo termine
- **Local Storage:** HTML5, maggiore capacità
- **Session Storage:** Dati temporanei nel browser

Esempio Pratico - Sistema di Login

```
<?php
// login.php - Gestione login con cookie "Ricordami"

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $username = $_POST['username'];
    $password = $_POST['password'];
    $remember = isset($_POST['remember']);

    // Verifica credenziali (esempio semplificato)
    if ($username == 'admin' && $password == 'pass123') {

        // Cookie di sessione standard
        setcookie('logged_in', 'true', 0, '/', '', true, true);
        setcookie('username', $username, 0, '/', '', true, true);

        // Se "Ricordami" selezionato, cookie persistente
        if ($remember) {
            $expire = time() + (86400 * 30); // 30 giorni
            setcookie('remember_token',
                hash('sha256', $username . time()),
                $expire, '/', '', true, true);
        }

        header('Location: dashboard.php');
        exit;
    }
}
?>
```

Concetti Chiave

- 1 **HTTP è Stateless:** Non mantiene stato tra richieste
- 2 **Cookie:** File di testo sul client per mantenere stato
- 3 **setcookie():** Funzione PHP per creare cookie
- 4 **\$_COOKIE:** Array per leggere cookie
- 5 **Scadenza:** Gestibile con timestamp Unix
- 6 **Sicurezza:** Usare Secure e HttpOnly
- 7 **Normativa:** Distinguere tecnici vs profilazione
- 8 **Limitazioni:** Modificabili dall'utente, 4KB max
- 9 **Alternative:** Sessioni PHP per dati sensibili

Ricorda

Usa i cookie per dati non sensibili e considera sempre la privacy degli utenti