Cofinanziato
dall'Unione europea

Corso PHP

Sessioni

PHP: Gestione delle Sessioni

La gestione delle sessioni in PHP è una funzionalità fondamentale per creare applicazioni web dinamiche. Una sessione permette di mantenere le informazioni dell'utente tra diverse richieste HTTP, consentendo, ad esempio, di implementare sistemi di autenticazione e carrelli di acquisto.

1. Cos'è una Sessione?

Una sessione è un modo per memorizzare informazioni (come le credenziali di accesso) su un utente durante la sua interazione con un'applicazione web. A differenza dei cookie, le sessioni non memorizzano i dati direttamente sul client (nel browser), ma su un server. Solo un identificatore di sessione (un ID di sessione) viene inviato al client.

2. Inizializzazione di una Sessione

Per iniziare a utilizzare le sessioni in PHP, è necessario chiamare la funzione `session_start()`. Questa funzione deve essere la prima cosa eseguita nello script, prima di qualsiasi output HTML, simile ai cookie.

Esempio:

```
<?php
session_start();
?>
```

Quando `session_start()` viene chiamata, PHP:

1. Verifica se una sessione esistente è già stata avviata (tramite l'ID di sessione inviato dal client).
2. Se non esiste una sessione, ne crea una nuova.
3. Associa un ID di sessione univoco, che viene inviato al client come cookie.

3. Salvataggio dei Dati in una Sessione

Una volta avviata la sessione, è possibile memorizzare informazioni utilizzando la superglobale `$_SESSION`, che è un array associativo.

Esempio:

```
$_SESSION['username'] = 'Mario';
$_SESSION['logged_in'] = true;
```

In questo esempio, due valori vengono salvati nella sessione: il nome utente e lo stato di login.

4. Accesso ai Dati di una Sessione

I dati salvati in una sessione possono essere recuperati in qualsiasi pagina dello stesso sito web (sempre dopo aver avviato la sessione con `session_start()`).

Esempio:

```php
session_start();
if(isset($_SESSION['username'])) {
    echo "Ciao, " . $_SESSION['username'];
} else {
    echo "Utente non autenticato";
}
```

5. Modifica dei Dati di una Sessione

Modificare i dati della sessione è semplice come aggiornare un valore in un array.

Esempio:

```
$_SESSION['username'] = 'Luigi';
```

6. Cancellazione dei Dati di una Sessione

Per rimuovere un singolo valore da una sessione, si usa `unset()` sulla chiave dell'array `$_SESSION`.

Esempio:

```php
unset($_SESSION['username']);
```

Per rimuovere tutti i dati della sessione ma mantenere la sessione attiva, si usa `session_unset()`.

Esempio:

```
session_unset();
```

7. Distruzione di una Sessione

Quando si desidera terminare completamente una sessione (ad esempio, al momento del logout), si deve usare `session_destroy()`. Questo non cancella l'array `$_SESSION` finché lo script non termina, ma invalida l'ID di sessione, rimuovendo la sessione lato server.

Esempio:

```
session_start();
session_destroy();
```

Per una rimozione completa, è buona pratica anche cancellare i cookie associati alla sessione.

Esempio di Logout Completo:

```
session_start();
session_unset();
session_destroy();
setcookie(session_name(), '', time() - 3600, '/');
```

8. Configurazione delle Sessioni

PHP offre varie opzioni di configurazione per le sessioni, che possono essere impostate tramite il file `php.ini` o tramite le funzioni `ini_set()`.

- **session.name**: Nome del cookie di sessione (di default è `PHPSESSID`).
- **session.save_path**: Percorso dove vengono salvati i file di sessione lato server.
- **session.gc_maxlifetime**: Tempo di vita (in secondi) dopo il quale i dati di sessione inutilizzati vengono eliminati.

Esempio di Configurazione:

```
ini_set('session.gc_maxlifetime', 3600); // Sessione valida per un'ora
```

9. Sessioni Sicure

Per garantire la sicurezza delle sessioni, è consigliabile adottare alcune misure:

- **Sessione via HTTPS**: Assicurarsi che il cookie di sessione venga inviato solo su connessioni sicure.

```
ini_set('session.cookie_secure', 1);
```

- **Impedire l'accesso via JavaScript ai cookie di sessione**:

```php
ini_set('session.cookie_httponly', 1);
```

- **SameSite Cookies**: Per prevenire attacchi CSRF, si può impostare il SameSite attribute:

```
ini_set('session.cookie_samesite', 'Strict');
```

- **Rigenerazione dell'ID di sessione**: Per mitigare gli attacchi di session fixation, è utile rigenerare l'ID di sessione dopo il login:

```
session_regenerate_id(true);
```

10. Sessioni Personalizzate

PHP permette anche di gestire le sessioni in modo personalizzato, utilizzando handler personalizzati per il salvataggio dei dati (es. in un database) o per la gestione del ciclo di vita delle sessioni. Questo viene fatto tramite la funzione `session_set_save_handler()`.

Conclusione

La gestione delle sessioni in PHP è uno strumento potente che permette di mantenere lo stato tra diverse richieste HTTP, essenziale per costruire applicazioni web interattive e sicure. È fondamentale capire le migliori pratiche di sicurezza e configurazione per proteggere le informazioni sensibili degli utenti e garantire un'esperienza utente fluida.

Gestione dei tipi di dato

In una sessione PHP, puoi salvare una vasta gamma di tipi di dati. La superglobale `$_SESSION` è un array associativo che può contenere diversi tipi di valori. Ecco i principali tipi di elementi che puoi salvare in una sessione:

1. Stringhe

Le stringhe sono uno dei tipi di dati più comuni memorizzati in sessione, utilizzate per conservare valori come nomi utente, messaggi, token di sicurezza, ecc.

```
$_SESSION['username'] = 'MarioRossi';
$_SESSION['token'] = 'abc123xyz';
```

2. Numeri Interi e Decimali

Puoi memorizzare numeri interi o decimali in una sessione, utili per contatori, punteggi, o valori numerici generali.

```
$_SESSION['user_id'] = 42;
$_SESSION['balance'] = 1234.56;
```

3. Booleani

Valori booleani (`true` o `false`) possono essere utilizzati per tenere traccia dello stato di qualcosa, come se un utente è autenticato o meno.

```
$_SESSION['logged_in'] = true;
$_SESSION['is_admin'] = false;
```

4. Array

Gli array possono essere memorizzati in sessione, permettendo di conservare collezioni di dati correlati. Possono essere array semplici o multidimensionali.

```php
$_SESSION['cart'] = ['item1', 'item2', 'item3'];
$_SESSION['user_data'] = [
    'name' => 'Mario Rossi',
    'email' => 'mario.rossi@example.com'
];
```

5. Oggetti

Anche gli oggetti possono essere salvati in una sessione, permettendo di mantenere lo stato di oggetti complessi tra le richieste. È importante notare che se salvi oggetti, devi garantire che le relative classi siano caricate correttamente nelle pagine dove accedi alla sessione.

```php
class User {
    public $name;
    public $email;

    public function __construct($name, $email) {
        $this->name = $name;
        $this->email = $email;
    }
}

$_SESSION['user'] = new User('Mario Rossi', 'mario.rossi@example.com');
```

6. Null

Puoi anche memorizzare valori `null` in una sessione. Questo può essere utile per indicare uno stato non definito o una mancanza di dati.

```
$_SESSION['last_activity'] = null;
```

Considerazioni Importanti

- **Dimensione dei dati:** Anche se è possibile salvare molti tipi di dati in sessione, è importante essere consapevoli delle dimensioni dei dati memorizzati, poiché sessioni molto grandi possono rallentare le prestazioni.

- **Serializzazione:** PHP serializza automaticamente gli array e gli oggetti quando li memorizza nella sessione. Tuttavia, è importante assicurarsi che le classi degli oggetti siano caricate correttamente quando si ripristinano dalla sessione.

- **Sicurezza:** Evita di memorizzare dati sensibili in sessione senza le dovute precauzioni, come crittografia o altre misure di sicurezza.

Salvare diversi tipi di dati in una sessione è molto utile per mantenere lo stato di un'applicazione tra le varie richieste dell'utente, migliorando l'esperienza utente e facilitando la gestione delle informazioni.