Sessions, authentification, et autorisation

Lien vers le cours

L. Delafontaine, avec l'aide de GitHub Copilot.

Retrouvez plus de détails dans le support de cours

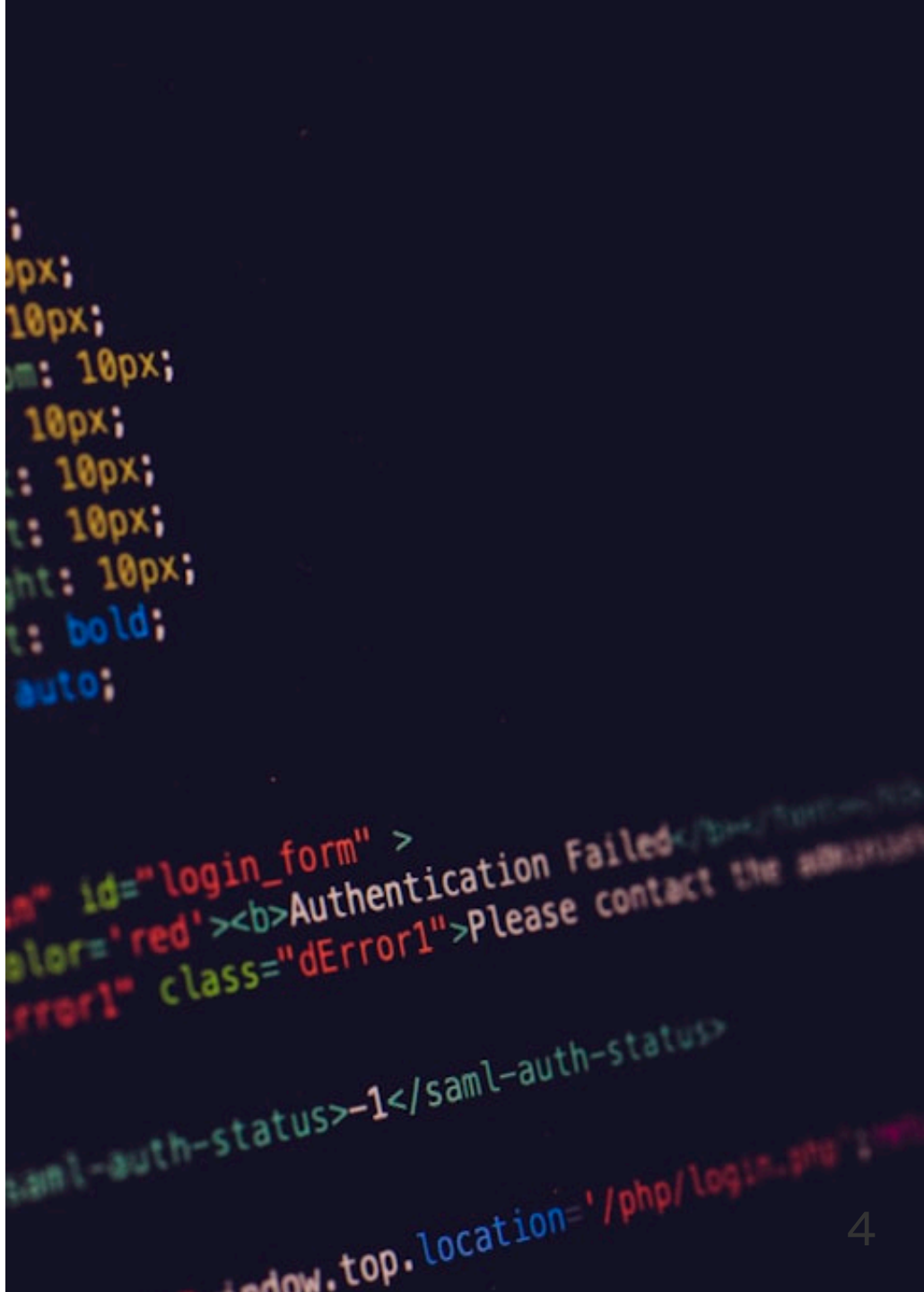
Cette présentation est un résumé du support de cours. Pour plus de détails, consultez le support de cours.

Objectifs

- Utiliser les sessions pour stocker des informations utilisateur.
- Différencier les sessions et les cookies.
- Implémenter l'authentification des utilisateur.trices à l'aide de sessions.
- Gérer l'autorisation des utilisateur.trices en fonction de leurs rôles.

Les sessions

- Permettent de stocker de l'information côté serveur.
- Identifiées par un identifiant de session unique (généralement stocké dans un cookie, renvoyé automatiquement à chaque requête).
- Utilisées pour maintenir l'état entre les requêtes HTTP (ex. : authentification).



Démarrer une session en PHP

Utiliser `session_start()` au début de chaque script PHP qui utilise des sessions :

```php
<?php
// Démarre la session
session_start();

// Stocke des informations dans la session
$_SESSION['user_id'] = 123;
$_SESSION['username'] = 'johndoe';
```

Récupérer des informations de session en PHP

Utiliser la superglobale `$_SESSION` pour accéder aux données de session :

```php
<?php
// Démarre la session
session_start();

// Récupère des informations de la session
$user_id = $_SESSION['user_id'] ?? null;
$username = $_SESSION['username'] ?? 'Invité';
```

Durée de vie des sessions

- Durée de vie courte définie par le serveur (24 minutes par défaut).
- Peut être configurée dans le fichier de configuration `php.ini`, mais ce n'est pas conseillé.
- Dans le futur, d'autres mécanismes plus robustes seront vus pour gérer la persistance des sessions.

Détruire une session en PHP

Utiliser `session_destroy()` pour détruire une session et supprimer toutes les données associées :

```php
<?php
// Démarre la session
session_start();

// Détruit la session
session_destroy();
```

Différences entre sessions et cookies

	Cookies	Sessions
Stockage	Côté client (navigateur)	Côté serveur - cookie pour identifier la session.
Sécurité	Moins sécurisés pour les informations sensibles	Plus sécurisées pour les informations sensibles
Taille	Limitée (environ 4 Ko par cookie)	Limitée par la mémoire serveur
Durée de vie	Configurable	Courte par défaut

Authentification et autorisation

- Les sessions permettent de gérer l'authentification et l'autorisation.
- **Concepts essentiels mais difficiles à implémenter correctement ! Dans de futurs unités d'enseignement, nous verrons des solutions plus robustes.**

Authentification

- Processus de vérification de l'identité d'un utilisateur.
- Généralement via un nom d'utilisateur et un mot de passe.
- Si les informations sont correctes (= authentification réussie), stocker l'état de connexion dans la session.

Autorisation

- Processus de vérification des droits d'accès d'un utilisateur.
- Détermine les actions qu'un utilisateur peut effectuer.
- Basé sur les rôles et permissions définis dans l'application.

Gérer l'authentification et l'autorisation avec des sessions (1/2)

Ce processus peut être résumé par les étapes suivantes :

1. Démarrer une session lors de la connexion de l'utilisateur.
2. Stocker l'ID de l'utilisateur et son rôle dans la session.
3. Vérifier les informations d'identification de l'utilisateur lors de la connexion.
4. Utiliser les informations de session pour autoriser ou interdire l'accès à certaines ressources.

Gérer l'authentification et l'autorisation avec des sessions (2/2)

```php
<?php
// Démarre la session
session_start();

// Vérifie si l'utilisateur est authentifié
if (!isset($_SESSION['user_id'])) {
    // Redirige vers la page de connexion
    header('Location: login.php');
    exit();
}
```

```php
// Vérifie les droits d'accès de l'utilisateur
if ($_SESSION['role'] !== 'admin') {
    // Refuse l'accès avec une erreur 403 Forbidden
    http_response_code(403);
    exit();
}

// L'utilisateur est authentifié
// et autorisé à accéder à la ressource
// ...
```

Un exemple plus complet est disponible dans les exemples de code.

Stocker les mots de passe de manière sécurisée dans la base de données (1/3)

- Les sessions permettent de mettre en place des mécanismes d'authentification et d'autorisation.
- Les données sensibles, comme les mots de passe, doivent être stockées de manière sécurisée.
- Hasher un mot de passe signifie le transformer en une chaîne de caractères fixe, difficile à inverser (`m0n-m0t-de-p4ss3` > `5f4dc...`).
- Utiliser des fonctions de hachage sécurisées, comme `password_hash()` et `password_verify()` en PHP (cf. exemple).

Stocker les mots de passe de manière sécurisée dans la base de données (2/3)

```php
<?php
// register.php
// Connexion à la base de données
$pdo = new PDO('mysql:host=localhost;dbname=mydatabase', 'username', 'password');

// Récupère les données du formulaire
$username = $_POST['username'];
$password = $_POST['password'];

// Hache le mot de passe
$hashedPassword = password_hash($password, PASSWORD_DEFAULT);

// Insère l'utilisateur dans la base de données
$stmt = $pdo->prepare('INSERT INTO users (username, password) VALUES (:username, :password)');
$stmt->execute(['username' => $username, 'password' => $hashedPassword]);
```

Stocker les mots de passe de manière sécurisée dans la base de données (3/3)

```php
<?php
// login.php
// Connexion à la base de données
$pdo = new PDO('mysql:host=localhost;dbname=mydatabase', 'username', 'password');

// Récupère les données du formulaire
$username = $_POST['username'];
$password = $_POST['password'];

// Récupère l'utilisateur de la base de données
$stmt = $pdo->prepare('SELECT * FROM users WHERE username = :username');
$stmt->execute(['username' => $username]);
$user = $stmt->fetch();
```

```php
// Vérifie le mot de passe
if ($user && password_verify($password, $user['password'])) {
    // Authentification réussie - démarre la session
    session_start();

// Stocke les informations utilisateur dans la session
    $_SESSION['user_id'] = $user['id'];
    $_SESSION['username'] = $user['username'];

    // Redirige vers la page d'accueil
    header('Location: index.php');
    exit();
} else {
    // Authentification échouée
    echo 'Nom d\'utilisateur ou mot de passe incorrect.';
}
```

Conclusion

- Les sessions permettent de maintenir l'état entre les requêtes HTTP à l'aide d'un cookie.
- Utiliser les sessions pour gérer l'authentification et l'autorisation des utilisateur.trices.
- Stocker les mots de passe de manière sécurisée en les hachant avant de les stocker dans la base de données.

Questions

Est-ce que vous avez des questions ?

À vous de jouer !

- (Re)lire le support de cours.
- Explorer les exemples de code.
- Faire les exercices.
- Poser des questions si nécessaire.

➡️ Lien vers le cours

N'hésitez pas à vous entraidez si vous avez des difficultés !

Sources

- Illustration principale par Richard Jacobs sur Unsplash
- Illustration par Aline de Nadai sur Unsplash
- Illustration par Markus Spiske sur Unsplash
- Illustration par Jon Tyson sur Unsplash
- Illustration par CDC sur Unsplash
- Illustration par Imre Tomosvari sur Unsplash
- Illustration par Nikita Kachanovsky sur Unsplash