Atelier développement

X75 : Développeur web front end (Partie 3)

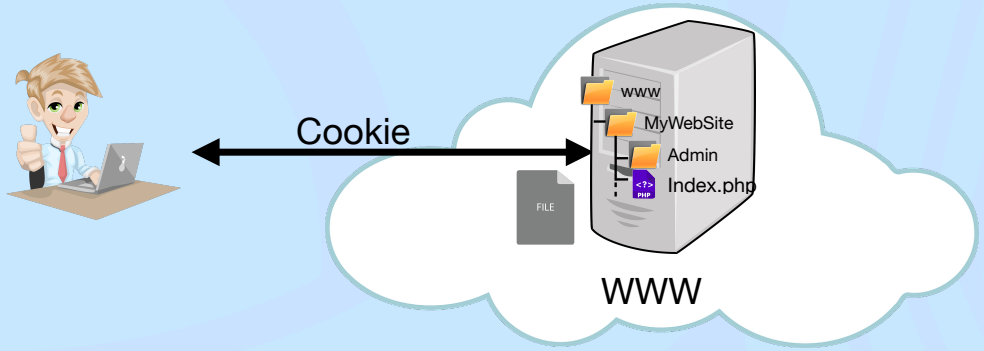
Atelier développement, bases

2025

Le BackOffice

Cookie et sessions

Cookie



- 1 ou plusieurs cookies = 1 Fichier texte enregistré par le navigateur web sur l'ordinateur de l'utilisateur
- Sauvegarde temporaire ou durable d'informations
- Les cookies sont échangés entre le client et le serveur

Consentement Explicite

- Création du cookie (php): setcookie(nom_cookie, valeur, **DuréeDeVie**, …);
  - Cet appel doit apparaître ***AVANT*** la balise <html>

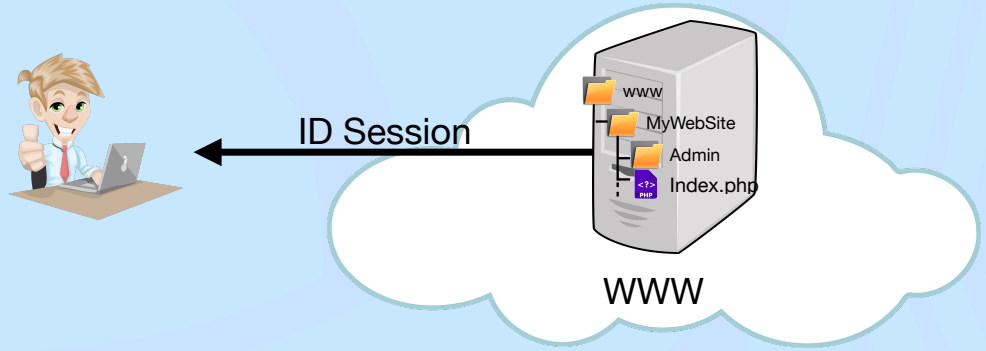
Valeur en seconde
par défaut = 0 = Durée de la session

- Utilisation de la superglobale $_COOKIE

```
if (isset($_COOKIE["nom_cookie"]))
   echo 'Le cookie existe ' . $_COOKIE["nom_cookie"] . '!<br />';
else
   echo 'Le cookie n\'existe pas <br />';
```

Cookie et sessions

Session



- Sauvegarde temporaire d'informations côté serveur pendant la visite d'un utilisateur.
- Chaque utilisateur obtient un identifiant de session unique généralement transmis via un cookie spécifique PHPSESSID.
- Démarrer une session (php): session_start()
  - la session se termine et est détruite à la fermeture du navigateur ou appel de session_destroy();
- La superglobale $_SESSION est accessible durant la session

```
sessiont_start();
if (isset($_SESSION[‘username'])) {
echo "Bienvenue, " . htmlspecialchars($_SESSION[‘username']);
}
$_SESSION['username'] = "John Doe";
```

Cookie et sessions

Problème

- Dans le cadre du développement d'un site web, il est nécessaire de mettre en place un système d'authentification permettant aux utilisateurs de se connecter avec un nom et un mot de passe.
- Ce système doit permettre:
  - à l'utilisateur de retrouver automatiquement le dernier nom utilisé lors de sa précédente connexion
  - s'assurer que son état de connexion soit affiché sur toutes les pages tant qu'il est connecté.

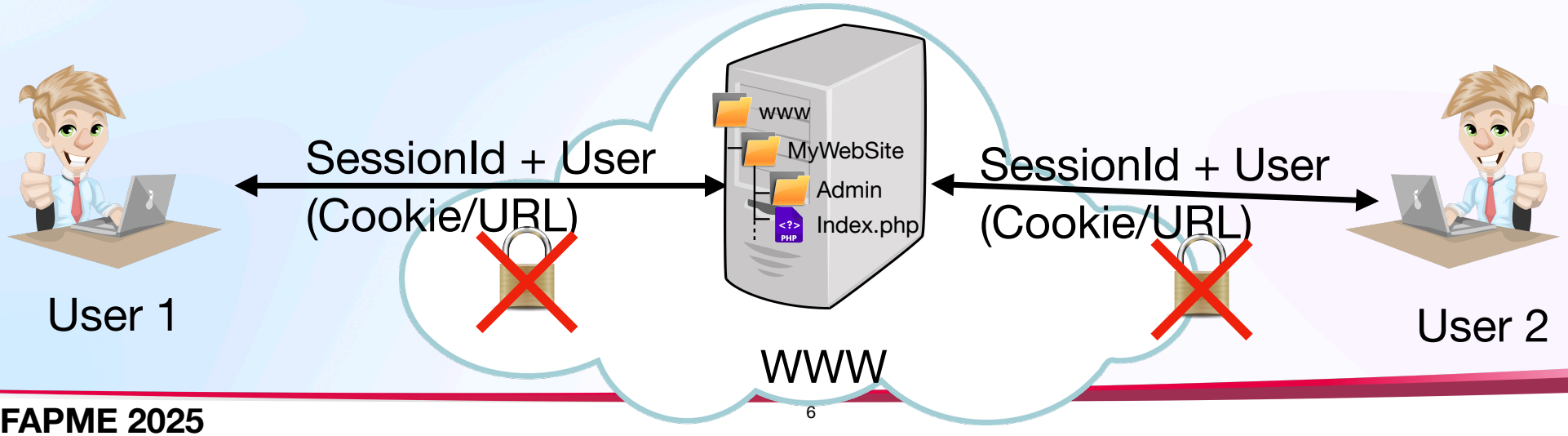
Création d'un blog

Le BackOffice

Constitué de pages web uniquement accessibles aux utilisateurs autorisés

Utilisateur autorisé = utilisateur différent des autres et qui a les droits

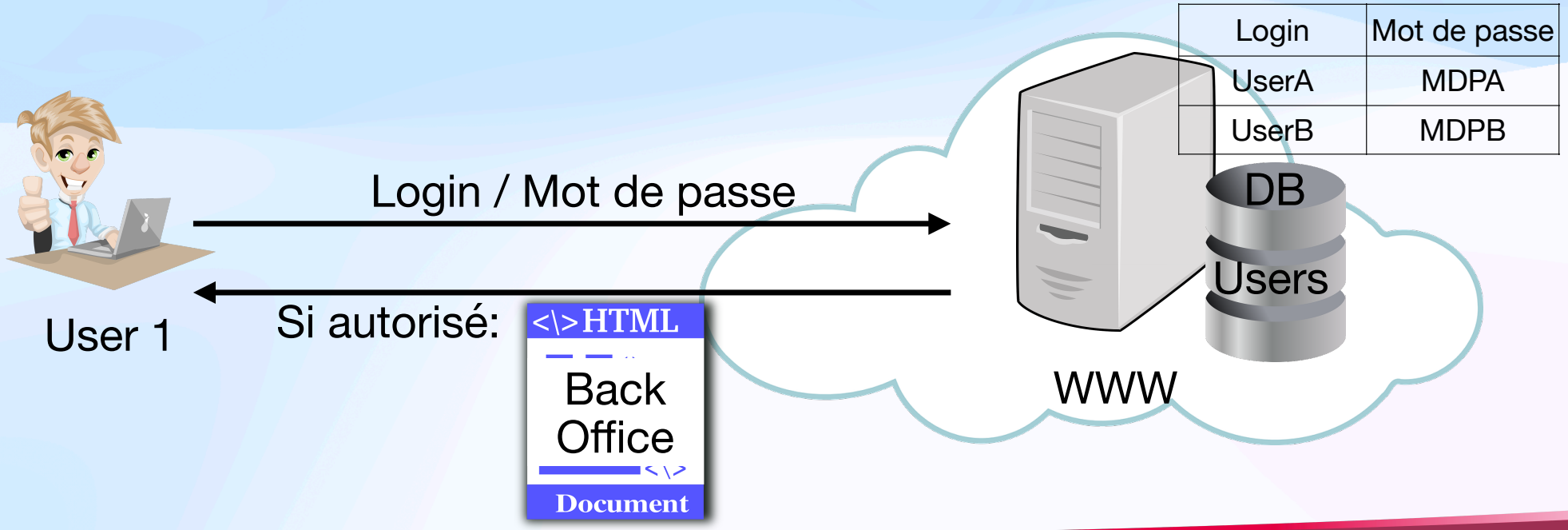
☑ assigner un identifiant unique à l'utilisateur = identifiant de session



Création d'un blog

Le BackOffice

☑ Vérifier si l'utilisateur à les droits = Vérifier en DB dans la table des utilisateurs



Sécurisation des données

Balise script

Insertion de script dans la page html

- Dans la balise <head> ou <body>

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <script>
        alert("Hello, World!");
    </script>
</head>
<body>
    <h1>Ma page</h1>
</body>
</html>
```

- En fin de <body> (améliore le chargement de la page)

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <title>Exemple</title>
</head>
<body>
    <h1>Ma page</h1>
    <script>
        console.log("Le script s'exécute après le chargement de la page. »);
    </script>
</body>
</html>
```

- Via un fichier externe (facilite la maintenance)

```
<!DOCTYPE html>
<html lang="fr">
<head>
   <script src="script.js"></script>
</head>
<body>
   <h1>Ma page</h1>
</body>
</html>
```

- Fichier script.js

```
console.log("Le script externe est bien chargé ! »);
```

Les iframe

Insérer une page web dans une autre

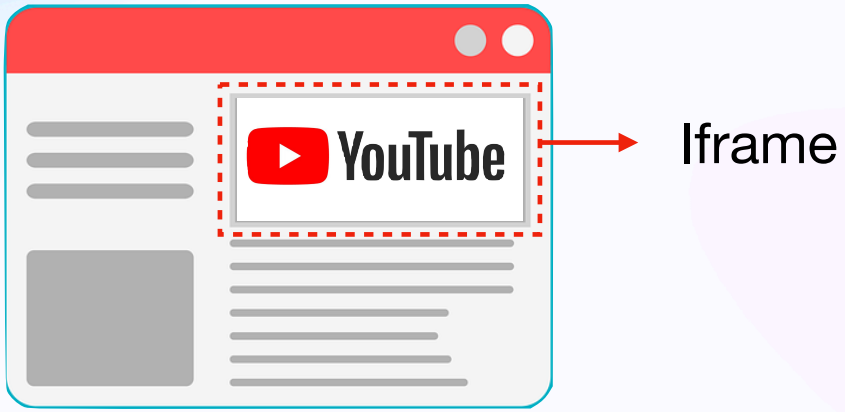
- L'élément <iframe> permet d'intégrer une autre page web dans une page HTML. Il est souvent utilisé pour afficher des vidéos, des cartes interactives, ou des contenus externes comme des documents.
- Intégration dans le HTML

<iframe src="https://www.example.com" width="600" height=« 400" sandbox></iframe>

L'attribut sandbox désactive l'exécution de scripts et d'autres actions potentiellement dangereuses.

Exemple CSS:

```
iframe {
    position: absolute;
    top: 100px;
    left: 500px;
    width: 200px;
    height: 80px;
    opacity: 1;   ⟵ Rendu invisible
    z-index: 2;   ⟵ Ordre Z des composants
}
```



Sécurisation des données

Mot de passe crypté dans la base de données

- Pour hacher le mot de passe pour le mettre en DB

```
$hashed_password = password_hash($password, PASSWORD_BCRYPT);
```

Hash du mot de passe contenant des méta données

Algorithme de hachage

- Pour comparer le mot de passe à celui qui est en DB:

```
password_verify($password, $user['password'])
```

Mot de passe saisi

Mot de passe en DB

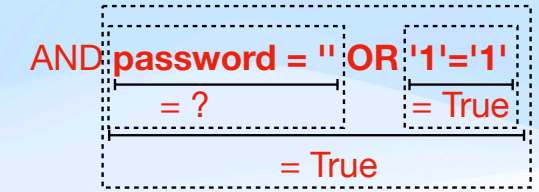
Retourne True si les mots de passe correspondent ou False

Sécurisation des données

Vulnérabilité injection SQL (SQLi)

- **Bypass d'authentification** (Formulaire vulnérable)

```
$query = "SELECT * FROM users WHERE email = '$email' AND password = '$password' " ;
$result = $mysqli->query($query);
```



- Injection des données: email: admin@exemple.com et mot de passe: **' OR '1' = '1'**
- Résultat: La connection est faite quelque soit le mot de passe
- Protection: Requêtes préparées et mot de passe crypté

```
$stmt = $mysqli->prepare("SELECT * FROM users WHERE email = ? AND password = ?");
$stmt->bind_param("ss", $email, $password);
$stmt->execute();
$result = $stmt->get_result();
if ($result->num_rows > 0) {
    echo "Connexion réussie";
} else {
    echo "Échec de la connexion »;
}
```

Deux s car les 2 arguments sont de type chaîne de caractères

- Dans une requête préparée, les arguments ne sont pas interprétés comme du code SQL

Sécurisation des données

Vulnérabilité Cross-Site Scripting (XSS)

- **Exécution d'un script côté client** (mise en base de données vulnérable)

  Affichage de saisie utilisateur sauvegardée dans la base de données (code vulnérable)

  ```
  echo "<p>Commentaire: " . $_POST['comment'] . " </p>";
  ```

- Saisie malveillante comme commentaire: `<script>alert('Exécution de code!');</script>`
- Résultat: Lorsqu'un utilisateur affiche la page contenant ce commentaire, le script s'exécute
- Protection: Utiliser htmlspecialchars($_POST['comment'], ENT_QUOTES, 'UTF-8')

Convertit les ' en "

Spécifie encodage

Sécurisation des données

Vulnérabilité Cross-Site Request Forgery (CSRF)

- **Suppression dans la base de données** (Vulnérabilité dans l'utilisation de données)

```
if (isset($_GET['delete']) && $_GET['confirm'] == 1) {
    $stmt = $mysqli->prepare("DELETE FROM users WHERE id = ?");
    $stmt->bind_param("i", $_SESSION['user_id']);
    $stmt->execute();
}
```

- Envoi d'un email malveillant alors que l'utilisateur est connecté:

```
<a href="https://site-victime.com/delete?confirm=1">Cliquez ici pour un cadeau 🎁</a>
```

- Résultat: L'utilisateur est supprimé de la base de données s'il clique sur le lien
- Protection:

  Générer un token CSRF unique s'il n'existe pas:

```
if (!isset($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
```

Génère 32 octets de données cryptographiquement sécurisés

Convertit les 32 octets binaires en une chaîne hexadécimale

  Mettre le token dans un champ caché pour y avoir accès:

```
<input type="hidden" name="csrf_token" value="<?php echo $_SESSION['csrf_token']; ?>">
```

  Tester si le token correspond:

```
if (!isset($_POST['csrf_token']) || $_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    die("⚠️ CSRF détecté ! »);
}
```

Sécurisation des données

Brute force sur un formulaire de connexion

- **Usurpation d'identité** (Vulnérabilité dans le code)

```
if ($user && password_verify($_POST['password'], $user['password'])) {
    echo "Connexion réussie »;
}
```

- Tentatives de mots de passe en bash:

```
for p in $(cat wordlist.txt); do
    curl -X POST -d "email=admin@example.com&password=$p" https://site-victime.com/login;
done
```

- Résultat: L'utilisateur se connecte en ayant trouvé le mot de passe

- Protection: Limiter le nombre d'essais (valeur en Session ou en DB)

```
$_SESSION['failed_attempts']++;
…
if ($_SESSION['failed_attempts'] >= 5) {
    die("❌ Trop de tentatives ! Réessayez dans quelques minutes.");
}
```

Sécurisation des données

Accès aux fichiers sensibles

- **Fichier contenant des informations de connexions,…:** .env, config.php,.git,…
- Solution : Pour apache, fichier de configuration .htaccess de configuration on rajoute:

```
<FilesMatch "(\.env|config\.php)$">
    Order allow,deny
    Deny from all
</FilesMatch>
```

Les règles Allow/Deny sont évaluées dans l'ordre
Pour le reste: Refuse tout
(Pour Apache > 2.4: Require all denied)

- **Accès à phpinfo.php, fichier de logs, répertoire d'upload,…**
- **"display_errors = On" en production dans php.ini**

Sécurisation des données

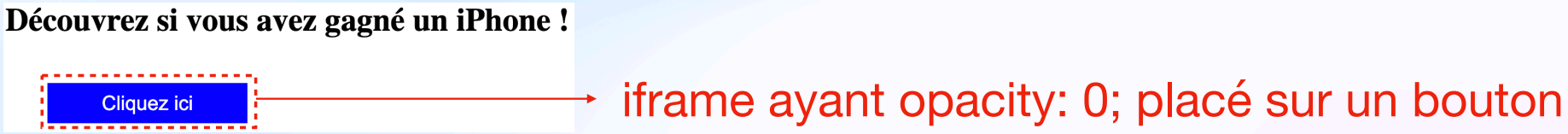
Clickjacking

- **Clickjacking**: Faire appuyer l'utilisateur sur une fenêtre qui n'est pas visible

  On place le site sur lequel on veut que l'utilisateur interagisse dans une iframe

  ```
  <iframe src="http://localhost/IFAPME/Vulnerability/siteausurper.php?email=hacker@gmail.com"></iframe>
  ```

  On place la frame en invisible sur la page à un endroit où l'utilisateur est incité à appuyer



- Solution:
  - en php ne pas autoriser la mise en iframe de la page: **header("X-Frame-Options: DENY »);**

header("Content-Security-Policy: frame-ancestors 'none';"); interdit intégration dans un iframe quelque soit l'origine

Sécurisation des données

Divers

- **Utilisation de librairies vulnérables**
- **Gestion des permissions dans la DB**
- **Manque de Journalisation**
- **RCE (Remote code execution)**
- **Redirection on sécurisée:**

```php
<?php
    $redirect_url = $_GET['url'] ?? 'default.php';

    header("Location: $redirect_url");
    exit;
?>
```

- **...**

Sécurisation des données

Validation côté client et côté serveur

- **Côté client: Attributs de validation,…**

  Contournables.

  - required (champ obligatoire)
  - min, max, minlength, maxlength
  - pattern
  - <input type="email", "number","url",...

Exemple:

```
<input type="text" name="nom" required pattern="[A-Za-z]+"
minlength="3" maxlength="10" title="Seules les lettres sont
autorisées" placeholder="Nom (lettres uniquement) »>

<input type="number" name="age" min="18" max="99"
placeholder="Âge (18-99) »>
```

- **Côté serveur**

Empêche SQLi, XSS,…

  - isset(), true si variable existe et non nulle
  - empty(), true si variable existe ou vide
  - filter_var(), valide ou filtre une variable
  - preg_match(), teste expression régulière
  - htmlspecialchars(),
  - strip_tags(), retire les balises html/php
  - mysqli_real_escape_string(), protège contre une SQLi

Questions

Vulnérabilité

- Vous êtes un développeur web chargé de concevoir un blog interactif qui permet aux utilisateurs de poster des articles et de commenter.
  - Gestion des utilisateurs (CRUD)
  - Commentaires sous articles
  - Upload de fichiers pour un article
  - ...