3iLINGENIEURS </les sciences informatiques>		Année : 2020/2021
NOM : **Prénom** **Groupe :**	***Web – J. Grasset***	**15/03/2021**

Partie écrite – Sans connexion ni document

1. Sur un site web marchand, on dispose d'un formulaire de connexion classique (login/pass) et d'un formulaire de recherche de produits. Que préconisez-vous entre un envoi des données en get et un envoi des données en post pour chacun d'eux ? Justifiez précisément vos choix.

- **Formulaire de connexion (login/pass) :**
  - Utilisation de la méthode POST.
  - Justification : Les données sensibles, telles que les identifiants de connexion, sont transmises de manière sécurisée via POST, car elles ne doivent pas être exposées dans l'URL. POST offre également une meilleure sécurité car les données ne sont pas visibles dans l'historique du navigateur et ne sont pas facilement interceptables.

- **Formulaire de recherche de produits :**
  - Utilisation de la méthode GET.
  - Justification : GET est approprié pour les formulaires de recherche où les données ne sont pas sensibles et peuvent être partagées via des liens. De plus, l'utilisation de GET pour les recherches est conforme à la nature des requêtes HTTP, où les paramètres de recherche peuvent être inclus dans l'URL pour faciliter le partage des résultats de recherche ou la sauvegarde des favoris.

2. Expliquez le rôle et le fonctionnement des « tokens CSRF ».

CSRF :
amener l'utilisateur sur une page d'un site «malveillant»
adresser une requête HTTP via formulaire au site ciblé,
pour une action nécessitant une authentification : si la
session de l'utilisateur y est ouverte, la requête peut
être acceptée
Pour éviter, utilisation de tokens(anti-)CSRF (token= jeton) :
le serveur génère un tokenqu'il stocke en session
Chaque formulaire ou élément sensible du site contient ce
token(par exemple avec un «input hidden» dans les formulaires)
toute demande du client doit contenir ce token: le formulaire
«pirate» ne le connaît pas
Ce tokena une durée de vie courte (communément 10 minutes)

3. Expliquez **chacune** des **lignes 2, 5, 6, 7 et 8** du code ci-dessous.

```
1.  <?php
2.  $cnx = new PDO('mysql:host=localhost;dbname=baseDS', "anatole", "Toto963");
3.  $cnx->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
4.  $cnx->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
5.  $texteReq = "select id, nom from utilisateur";
6.  $req = $cnx->prepare($texteReq);
7.  $req->execute();
8.  $tabRes = $req->fetchAll();
```

- **Ligne 2 :** Cette ligne crée une nouvelle instance de la classe PDO, qui représente une connexion à la base de données. Les paramètres passés dans le constructeur spécifient le type de base de données (MySQL dans cet exemple), le nom d'hôte (localhost), le nom de la base de données (baseDS), le nom d'utilisateur (anatole) et le mot de passe (Toto963) pour se connecter à la base de données.

- **Ligne 5 :** Cette ligne définit une requête SQL sous forme de chaîne de caractères. Dans cet exemple, la requête sélectionne les colonnes "id" et "nom" de la table "utilisateur". Cette requête sera exécutée ultérieurement pour récupérer les données de la base de données.

- **Ligne 6 :** Cette ligne prépare la requête SQL définie précédemment pour son exécution. La méthode `prepare()` retourne un objet de type PDOStatement , qui représente la requête préparée prête à être exécutée.

- **Ligne 7 :** Cette ligne exécute la requête préparée. La méthode `execute()` exécute la requête SQL avec les éventuels paramètres fournis lors de la préparation de la requête. Dans ce cas, aucun paramètre supplément aire n'est requis car la requête ne contient pas de variables liées.

- **Ligne 8 :** Cette ligne récupère le résultat de l'exécution de la requête SQL sous forme d'un tableau associatif. La méthode `fetchAll()` retourne un tableau contenant toutes les lignes de résultat de la requête. Chaque élément du tableau est lui-même un tableau associatif contenant les valeurs des colonnes sélectionnées dans la requête.

4. On souhaite **afficher** le contenu de **$tabRes** (ligne 8 de la question précédente), en supposant que la requête a retourné plusieurs lignes de résultats.

Complétez le code ci-dessous pour que l'affichage produise une liste à puces (<ul>, <li>), avec dans chaque ligne l'id et le nom de l'utilisateur. (Répondez dans le code source).

```
8.  $tabRes = $req->fetchAll();
    foreach(                )
    {
    }
```

```
$tabRes = $req->fetchAll();
echo "<ul>";
foreach($tabRes as $row) {
    echo "<li>ID: " . $row['id'] . ", Nom: " . $row['nom'] . "</li>";
}
echo "</ul>";
```

5. Qu'est-ce qu'une requête asynchrone (« ajax ») ? Citez un exemple d'application.

Une requête asynchrone, ou AJAX, permet à une page web de communiquer avec le serveur en arrière-plan, sans recharger toute la page. Par exemple, sur un site de médias sociaux, les commentaires peuvent être chargés dynamiquement sans rafraîchir la page principale.