# Exercices : thème 2 - Question 3 (web)

## Question 3 (web) : La résolution de tous les problèmes de gestion est-elle automatisable ?

*Au travers de ces quelques exercices, nous allons étudier et concevoir ensemble vos premières pages web. A ce titre, nous allons découvrir comment définir leur contenu d'une part et leur mise en forme d'autre part. Ainsi, l'on fera la découverte des langages HTML et CSS, lesquels sont incontournables dans le cadre de la réalisation de sites internet, ou plus généralement dans le cadre du développement de logiciel faisant intervenir des technologies web. Quant à ces deux langages, il s'agit essentiellement d'en comprendre la syntaxe et de comprendre ce qu'ils sont et à quoi ils servent. Vos projets vous permettront de vous améliorer en la matière. Surtout, nous allons aborder la programmation PHP et faire resurgir les notions déjà abordées lorsque nous avons étudié la programmation VBA.*

## Exercice 1 : structure d'une page web et balises

| | |
|---|---|
| **HTML** | Le HTML est un langage permettant de décrire le contenu d'une page web et les ressources dont elle a besoin pour s'afficher correctement dans un navigateur. Les contenus et ressources : textes, images, feuilles de styles CSS, vidéo, son, etc. |
| **Balise** | Le HTML est un langage à balises, à savoir que les zones d'une page HTML sont délimitées par des éléments appelés marqueurs ou encore balises :<br>• Une balise ouvrante (exemple : ) marque le début d'une zone ;<br>• Une balise fermante (exemple : ) marque la fin d'une zone ;<br>• Si on a ouvert une balise, l'on doit la fermer ;<br>• Une balise peut être à la fois ouvrante et fermante (exemple : <br/>) ;<br>• Chaque balise a une signification et/ou un comportement qui lui est propre. |
| **Attribut** | Les balises HTML peuvent avoir des attributs. Ces derniers permettent en quelque sorte d'apporter une précision supplémentaire. Un attribut a un nom et une valeur. Les valeurs que peuvent prendre un attribut sont parfois prédéfinies, auquel cas il convient d'utiliser l'une des valeurs prédéfinies.<br>Exemple : <input name="unChamp" type="text" /><br>Commentaire : La balise input permet de définir un champ de formulaire. On a utilisé ses attributs name et type qui ont respectivement pour valeur unChamp et text. |

**Structure minimale d'une page web**

```
<!DOCTYPE html>
<html>
    <!-- en-têtes de la page HTML -->
    <head>
        <title>Titre de ma page web</title>
    </head>
    <!-- contenus de la page HTML -->
    <body>
        Le contenu de ma page web.
    </body>
</html>
```

**DOCTYPE :** permet de préciser au navigateur la version HTML utilisée, en l'occurrence HTML5.
**HTML :** permet de préciser le début et la fin de la page ;
**HEAD :** marque le début et la fin des en-têtes de la page. Les en-têtes permettent de préciser le titre de la page, de définir des métadonnées ou encore d'importer des ressources telles que feuilles de styles CSS ou script JavaScript ;
**BODY :** marque le début et la fin du corps de la page web, à savoir du contenu visible de la page web : textes, images, vidéos, etc.

Questions :

Dans le répertoire « C:/wamp/www », créer un dossier « tests ». A l'aide de NotePad++, créer puis enregistrer, dans le répertoire nouvellement créé, un fichier PHP « firstpage.php » contenant le code HTML minimal d'une page web.

**1.** Entre les balises <body> et </body>, ajouter le contenu textuel « Bonjour tout le monde ! ». Ceci fait, ouvrir le fichier dans le navigateur (« http://localhost/tests/firstpage.php »). Indiquer le résultat obtenu.

**2.** Rappeler le type d'architecture logicielle vous ayant permis de consulter cette ressource (« firstpage.php »). Quel protocole, quel nom de domaine et quelle URI avez-vous utilisés ?

**3.** Entre les balises <head> et </head>, ajouter <title>Ma première page WEB</title>. Actualiser la page web ouverte dans le navigateur. Quelle différence constatez-vous ?

**4.** Entre les balises <body> et </body>, ajouter le contenu suivant : Quelques<br/>tests<br/>HTML. Actualiser le navigateur. En déduire l'utiliser du marqueur br.

**5.** Entre les balises <body> et </body>, ajouter la ligne précisée ci-dessous. Actualiser. En déduire l'utilité du marqueur img.

```
<img src="https://fr.wikipedia.org/static/images/project-logos/frwiki.png" />
```

**6.** Ajouter l'attribut height de valeur "75" au marqueur img précédent, puis actualiser. En déduire l'utilité de cet attribut.

**7.** Dans le corps de la page web, ajouter les lignes suivantes, puis actualiser. En déduire le sens et/ou l'utilité de chacun des marqueurs.

```
<form>
    <label>Nom :</label> <input name="nom" type="text" placeholder="[Votre nom]" />
    <br/>
    <label>Né le :</label> <input name="anniv" type="date" />
    <br/>
    <label>Newsletter ?</label> <input name="news" type="checkbox" />
    <br/>
    <label>Statut :</label>
    <input name="status" type="radio" value="étudiant" /> Etudiant
    <input name="status" type="radio" value="professeur" /> Professeur
</form>
```

**8.** En faisant éventuellement varier les valeurs des attributs, déduire l'utilité et/ou le sens de chacun des attributs (nom et valeur(s)) du code HTML précédent.

**9.** Dans le corps de la page web, ajouter les lignes suivantes, indenter le code puis actualiser. En déduire l'utilité de chacun des marqueurs utilisés.

```
<table>
    <thead><tr><th>Nom</th><th>Né le</th><th>News</th><th>Statut</th></tr></thead>
    <tbody>
        <tr><td>Gregorio</td><td>11/11/1982</td><td>Oui</td><td>Professeur</td></tr>
        <tr><td>Doroté</td><td>29/02/1978</td><td>Oui</td><td>Professeur</td></tr>
        <tr><td>Camille</td><td>08/05/2000</td><td>Oui</td><td>Eleve</td></tr>
        <tr><td>Gertrude</td><td>08/05/2001</td><td>Oui</td><td>Eleve</td></tr>
```

```
    </tbody>
</table>
```

**10.** Entre les balises <tbody> et </tbody> de l'exemple précédent, insérer le code HTML suivant, l'indenter puis en déduire l'utilité de l'attribut colspan.

```
<tr><td colspan="2">2 en 1</td><td colspan="2">2 en 1</td></tr>
```

**11.** Entre les balises <body> et </body>, insérer le code HTML suivant, l'indenter puis en déduire l'utilité des balises ul et li.

```
<ul>
    <li>
        Texte 1
        <ul>
            <li>Texte 1.1</li><li>Texte 1.2</li><li>Texte 1.3</li>
        </ul>
    </li>
    <li>
        Texte 2
        <ul>
            <li>Texte 2.1</li><li>Texte 2.2</li><li>Texte 2.3</li>
        </ul>
    </li>
    <li>
        Texte 3
        <ul>
            <li>Texte 3.1</li><li>Texte 3.2</li><li>Texte 3.3</li>
        </ul>
    </li>
</ul>
```

## Exercice 2 : mise en forme d'une page web

| | |
|---|---|
| **CSS et Feuille de style** | Le **CSS** est un langage permettant de décrire la mise en forme de pages web. Il utilise la notion de sélecteur. Un **sélecteur** permet de préciser le contenu de la page web auquel un ensemble de **règles de mise en forme** vont s'appliquer.<br>Un fichier contenant du CSS est appelé une **feuille de styles**. |
| **Importer une Feuille de style** | Il existe 3 manières d'intégrer de la mise en forme CSS à une page web. Nous ne retiendrons qu'une méthode, la plus « propre des 3 », celle recommandée.<br>Le marqueur link est une balise que l'on place typiquement entre les balises <head> et </head>. Ce marqueur permet entre autres d'importer une feuille de style CSS à partir de son chemin d'accès (URL, URI, etc).<br>On procède comme suit : <link rel="stylesheet" href="/chemin/vers/fichier.css" />.<br>Les règles de mises en forme définies dans la feuille de styles seront dès lors appliquées au contenu de la page web. |

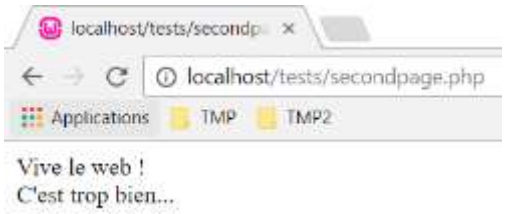
Quelques exemples de mises en forme vaudront mieux que de longs discours. On se donne deux fichiers, un fichier PHP appelé « secondpage.php » et une feuille de style CSS appelée « secondpage.css ».

On part du fichier « secondpage.php » suivant. Initialement, aucune règle de mise en forme n'est définie dans le fichier « secondpage.css ».

**Code HTML de « secondpage.php »**

```
<!DOCTYPE html>
<html>
 <head>
      <link rel="stylesheet" href="./une_page.css" />
 </head>
 <body class="laClasse">
      <div id="unId" class="uneClasse">
            <div class="texte1">Vive le web !</div>
            <div class="texte2">C'est trop bien...</div>
      </div>
 </body>
</html>
```

**Aperçu de « secondpage.php » (Navigateur)**




**Code CSS de « secondpage.css »**

```
html{
    height: 100%;
}
body{
    font-family: Arial;
    background: url(http://www.babelio.com/users/QUIZ_Le-quiz-special-geek-_7288.jpeg)
        no-repeat center center / contain;
}
.uneClasse{
    margin: auto; width: 200px; border: solid 3px #000000;
    padding: 32px; border-radius: 50%;
}
```

**Aperçu de « secondpage.php » (Navigateur)**

**Code CSS ajouté à « secondpage.css »**

```
.texte1, .texte2{
    text-align: center;
}
.texte1{
    font-weight: bold;
    margin-bottom: 6px;
}
.uneClasse:hover{
    cursor: pointer;
    background: #ffcc00;
}
```

**Aperçu de « secondpage.php » (Navigateur)**

Les langages HTML et CSS étant certes nécessaires à la réalisation de vos projets mais peu ou pas utiles à votre écrit de fin d'année, je vous laisse le soin d'effectuer des recherches, tutoriels et autres documentations, si vous souhaitez en savoir plus. Je reste également à l'écoute de vos éventuelles interrogations.

Questions :

**1.** Dans un premier temps, tester les deux précédents exemples. Expliquer l'utilité de chacun des sélecteur CSS. Expliquer ensuite l'utilité de chacune des propriétés CSS.

**2.** Dans un second temps, créer le fichier « firstpage.css ». A l'aide d'une balise link, charger cette feuille de style dans la page PHP « firstpage.php ». Finalement, proposer quelques améliorations de mise en forme à cette même page PHP. Expliciter les propriétés CSS que vous pourrez utiliser à cet effet.

Réponses :

**1.**

**2.**

## Exercice 3 : programmation PHP

### Partie 1 - Utiliser le langage PHP

Très bien, nous avons développé des pages PHP. Pourtant, nous n'avons pas encore programmé en PHP. Mais alors, il s'agit quand même de pages PHP ? Oui, car, comme nous allons le voir, le PHP vient s'intégrer au sein du contenu des pages web, à savoir au sein du HTML des fichiers « .php ».

Notions :

| | |
|---|---|
| **PHP** | Contrairement au HTML et au CSS, le PHP est bel et bien un langage de programmation. Il permet d'utiliser les notions que nous avons déjà étudiées en VBA : les conditions (if), les boucles (while, for, foreach), les variables, les tableaux, les fonctions et procédures, etc. |
| **Ajout du PHP** | Pour créer une page web PHP, il faut tout d'abord créer un fichier « .php » au sein duquel vous pouvez librement insérer du code HTML. Il convient, pour ajouter du PHP dans ce fameux fichier, d'utiliser les marqueurs <?php et ?> qui annoncent respectivement le début et la fin d'un « bout de code » PHP. Et c'est entre ces marqueurs que vous pouvez placer votre code PHP. |
| **Ecrire un contenu en PHP** | Outre le fait que vous puissiez directement écrire du contenu en HTML, PHP permet d'écrire dans la page (du texte, des nombres, le contenu d'une variable, etc.) :<br>`<?php`<br>`  // déclaration d'une variable contenant une chaîne de caractères`<br>`  $uneVariable = "Un texte dans une variable" ;`<br>`  echo "Ma variable contient : " . $uneVariable ;`<br>`?>`<br>*N.B. : le caractère « . » permet de concaténer des chaînes de caractères.* |
| **Inclure un autre contenu PHP** | Pour diverses raisons, vous pouvez avoir besoin d'inclure dans une page PHP le contenu d'un autre fichier PHP :<br>`<?php`<br>`  // inclusion du fichier « fichier_inclus.php » qui se trouve dans le même dossier`<br>`  include __DIR__ . "/fichier_inclus.php" ;`<br>`?>`<br>*N.B. : __DIR__ permet de récupérer le chemin vers le dossier courant.* |
| **Les variables** | Comme dans tout langage de programmation, PHP permet d'utiliser des variables, c'est-à-dire de créer des petits espaces mémoire ayant un nom, d'y stocker des valeurs et de faire des calculs avec celles-ci :<br>`<?php`<br>`  // Déclaration de variables (en PHP, préfixées par un $)`<br>`  $a = 5.50 ;`<br>`  $b = 6 ;`<br>`  $c = $a * $b ;`<br>`  echo $a . " x " .  $b . " = " . $c ;`<br>`?>` |

**Les tableaux**

Les tableaux sont des variables un peu particulières, lesquelles permettent de stocker plusieurs valeurs :

```
<?php
  // Déclaration d'un tableau contenant 4 éléments (4 valeurs)
  $unTableau = array(18, "2ème valeur", "valeur 3", 3.141592) ;
  // Affichage des 1er et 4ème éléments du tableau
  echo $unTableau[0]  . "<br/>"; // « [0] » permet d'utiliser le 0+1 = 1ère élément
  echo $unTableau[3]  . "<br/>"; // « [3] » permet d'utiliser le 3+1 = 4ème élément
  // Modification puis affichage du 4ème élément du tableau
  $unTableau[3] = 2.71828 ;
  echo $unTableau[3] . "<br/>";
  // Ajout d'un 5ème élément à la fin du tableau, puis affichage
  $unTableau[] = 3.141592 ; // « [] » permet d'ajouter un élément à la fin du tab.
  echo $unTableau[4] ;
 // Récupèration du nombre d'éléments du tableau, puis affichage
 $unNombre = count($unTableau) ;
 echo "Le tableau ''unTableau'' compte " . $unNombre . " élément(s). " ;
?>
```

**Remarque :** l'un des avantages (et inconvénients) de PHP, c'est qu'il n'est pas nécessaire de préciser le type de la variable. PHP est faiblement typé. Et une variable peut accepter n'importe quel type de valeur : nombre, chaîne, tableau, etc. En PHP, c'est le contenu de la variable qui en détermine le type.

Questions :

**1.** Tout d'abord, dans le répertoire « tests », créer le fichier « thirdpage.php » afin que nous puissions tester ensemble les précédents exemples.

**2.** Dans le même fichier, créer un tableau contenant les jours de la semaine, puis afficher les jours de la semaine à l'aide du tableau créé.

**3.** Afficher les jours de la semaine, mais cette fois-ci au moyen d'une boucle POUR (for) parcourant le tableau des jours de la semaine. Vous vous inspirerez d'un exemple trouvé sur internet.

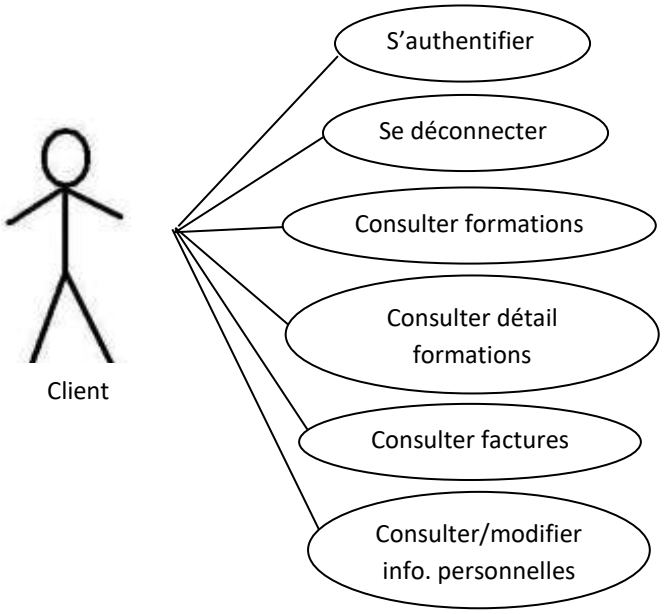
## Partie 2 - Interagir avec les utilisateurs en PHP

Contexte **:**

On se place désormais dans le « micro-contexte » suivant :

- La société Wissen souhaite mettre à la disposition de ses clients un espace client (des sociétés) disponible en ligne ;
- L'espace client permettra au client :
  - De consulter les formations qu'il a commandées et le détail de chaque formation : documents (devis, convention de formation, programme de formation, etc.), salariés participant, etc. ;
  - De consulter les factures qu'il a reçues ;
  - De consulter et modifier ses informations personnelles ;
  - De se déconnecter.
- Cette espace client est sécurisé au moyen d'une interface d'authentification au travers de laquelle un utilisateur s'identifie en saisissant son adresse mail et son mot de passe.

Le **diagramme de cas d'utilisation*** de l'espace client est de ce fait le suivant :




** Ce diagramme est utilisé pour décrire schématiquement les fonctionnalités d'un logiciel et les acteurs qui y ont accès. Vous devez être capables de produire de telles diagrammes, en particulier dans le cadre de votre projet.*

Pour le moment, notre attention portera seulement sur l'authentification et l'accès aux différentes pages de l'espace client. Nous n'étudierons pour l'instant pas la conception effective de cet espace client.

Notions **:**

| | |
|---|---|
| **Les paramètres GET** | Les paramètres GET sont transmis directement dans l'URL, exemple :<br>○ https://localhost/tests/thirdpage.php?unMessage=Hello World !&unNom=José<br>○ Ici, les paramètres sont « unMessage » et « unNom ». Ils sont séparés par un caractère « & » (prononcé « et commercial ») ;<br>○ Ils ont respectivement pour valeur « Hello World ! » et « José ».<br>PHP permet de récupérer ces paramètres grâce à la variable globale $_GET qui est ce qu'on appelle un tableau associatif car il permet de récupérer une donnée à partir d'un nom, appelé clef :<br><?php **echo** "Mon paramètre ''unMessage'' vaut : " . $_GET["unMessage"] ; ?> |

**Les paramètres POST**

Les paramètres POST ne sont pas transmis via l'URL. Ils sont particulièrement utilisés afin de récupérer le contenu des formulaires. On peut récupérer ces paramètres cette fois-ci grâce à la variable globale $_POST.

```
<form action="/tests/fourthpage.php" method="POST">
  <!-- Affichage de deux champ -->
  Adresse mail : <input name="email" type="text" /> <br/>
  Mot de passe : <input name="mdp" type="password" /> <br/>
  <!-- Bouton pour soumettre le formulaire -->
  <input type="submit" value="Se connecter" />
</form>
<br/>
<?php
  // Teste si les paramètres ''email'' et ''mdp'' existent
  echo "Paramètres POST reçu : <br/>" ;
  if(isset($_POST["email"], $_POST["mdp"])){
    // Affiche des deux paramètres transmis par l'utilisateur
    echo "- valeur du paramètre ''email'' : " . $_POST["email"] . "<br/>" ;
    echo "- valeur du paramètre ''mdp'' : " . $_POST["mdp"] ;
  }else{
    echo "encore rien..." ;
  }
?>
```

**Les paramètres de session**

Si les paramètres GET et POST n'existent que le temps d'une requête, les paramètres de session peuvent être utilisés toute la durée de vie d'une session. Ce qu'il faut retenir, c'est que :

- Ces paramètres peuvent être utilisés grâce à la variable globale $_SESSION et supprimés grâce à la fonction unset (exemple : unset($_SESSION["email"])) ;
- Ils permettent de conserver des données propres à l'utilisateur pendant plusieurs requêtes, en principe au moins tant que l'utilisateur ne ferme pas le navigateur.

Exemple :

```
<?php
  // A placer avant tout code HTML pour pouvoir utiliser les sessions.
  session_start() ;
  // Mise en session d'un paramètre ''email'' puis affichage
  $_SESSION["email"] = "quelquechose@gmail.com" ;
  echo $_SESSION["email"] ;
?>
```

**Les paramètres HTTP en PHP**

On crée souvent des pages web afin que les internautes puissent interagir avec le programme (exemple : formulaires). Pour cela entre autres, le protocole HTTP introduit la notion de paramètres. Cela signifie qu'il est possible pour le client (le navigateur) de transmettre des données au serveur (le serveur web) et ainsi à notre programme PHP. Parmi ces données pouvant être transmises, nous retiendrons en outre les paramètres GET et POST.

Questions :

Avant toute chose, effectuons quelques petits tests afin de bien comprendre ces notions de paramètres GET/POST et de paramètres de session.

**1.** Tout d'abord, dans le répertoire « tests », créer le fichier « fourthpage.php » afin que nous puissions tester ensemble les précédents exemples portant respectivement sur les paramètres GET, les paramètres POST et les paramètres de session.

A présent, copier-coller le dossier « wissen » dans votre répertoire : « C:/wamp/www/ ».

**2.** Expliquer le contenu du fichier « connexion.php ». En particulier, à partir de quel moment le paramètre de session « email » n'est-il plus vide ? En déduire l'utilité de la première condition.

**3.** Adapter le code PHP du fichier « connexion.php » de sorte que l'utilisateur ne puisse s'authentifier que si son identifiant est « joseph.fourier@yahoo.com » et son mot de passe « admin ». N.B. : en PHP, l'opérateur « ET » est « && ».

**4.** Compléter le code précédent de sorte que, lorsque l'utilisateur saisi est incorrect, un message d'erreur s'affiche : « Adresse mail et/ou mot de passe incorrect ». Le message d'erreur devra être affiché en-dessous du bouton « Se connecter ».

**5.** Améliorer la mise en forme du formulaire de connexion.

**6.** S'authentifier puis accéder aux pages « accueil.php » et « connexion.php » dans le navigateur. Est-il possible de se déconnecter ?

**7.** Ouvrer le fichier « accueil.php » sous NotePad++ et observer son résultat dans le navigateur. Le fichier « accueil.php » décrit-il tout le contenu de la page d'accueil ? Justifier.

**8.** Consulter le fichier « header.php » (en-tête) sous NotePad++. Quel est l'intérêt de la condition ?

**9.** Consulter les fichiers PHP préfixés par « mes_ » sous NotePad++. En déduire l'intérêt des fichiers « header.php » et « footer.php ». Qu'aurait-on dû faire si l'on n'avait pas utilisé la fonction « include » ?

**10.** A l'aide de la fonction « unset », compléter le code de la page « me_deconnecter.php » pour faire de sorte que l'utilisateur soit déconnecté.

## Partie 3 - Se connecter et manipuler une base de données en PHP

Très bien, nous sommes parvenus à nous connecter à notre espace client. Malheureusement, nous ne disposons que d'un seul compte utilisateur… Que faire si, et c'est certain, la société Wissen, a plus d'un Client ? Il nous faut désormais une base de données ! Pourquoi ? Pour stocker les comptes utilisateur… Et, dès lors qu'un utilisateur tentera de se connecter, il nous faudra alors vérifier si l'utilisateur existe. S'il existe, son mot de passe devra être correct. S'il l'est, la connexion doit normalement avoir réussi. Miracle ! Notre espace client sera désormais accessible à plusieurs utilisateurs.

Notions :

| | |
|---|---|
| **Se connecter à une base en PHP** | Afin de pouvoir exécuter une requête SQL (SELECT, INSERT, UPDATE, DELETE, etc.), il faut d'abord se connecter au serveur de bases de données. Pour ce faire, PHP met à votre disposition la fonction suivante :<br>○ mysqli_connect($host, $user, $password)<br>○ *Paramètres :* $host<br>- $host : l'adresse (URL) permettant d'accéder au serveur de base de données ;<br>- $user : le nom de l'utilisateur ;<br>- $password : le mot de passe de l'utilisateur.<br>○ *Valeur de retour :* une connexion à la base de données.<br>Exemple :<br>`<?php`<br>`  $connexion = mysqli_connect("localhost", "root", "") ;`<br>`?>` |
| **Sélectionner une base en PHP** | Une fois la connexion au SGBD récupérée, pour manipuler une base de données, il faut encore sélectionner la bonne base :<br>`<?php`<br>`  mysqli_select_db($connexion, "db_gest_wissen") ;`<br>`?>` |
| **Exécuter une requête en PHP** | Nous pouvons désormais exécuter des requêtes ainsi qu'en récupérer le résultat. Pour ce faire, PHP met en outre à votre disposition la fonction suivante :<br>○ mysqli_query($connection , $query)<br>○ *Paramètres :* $host<br>- $query : la requête SQL sous forme de chaîne de caractères ;<br>- $connection : la connexion récupérée à l'aide de la fonction mysqli_connect(…).<br>○ *Valeur de retour :*<br>- s'il s'agit d'une requête SELECT, la fonction retourne le résultat de la requête ou false si la requête échoue du fait d'une erreur. Il peut être exploité entre autres grâce aux fonction mysqli_fetch_assoc($result) et mysqli_num_rows($result) ;<br>- s'il s'agit d'une requête INSERT, UPDATE ou encore DELETE, la fonction retourne true ou false selon si la requête réussit ou échoue. On peut ensuite utiliser la fonction mysqli_affected_rows() pour connaître le nombre de lignes affectées par la requête mais aussi mysqli_insert_id() pour connaître le dernier identifiant auto-incrémenté en cas de requête INSERT. |

**Exemple de requête en PHP**

```php
<?php
  // On récupère une connexion au SGBD
  $connexion = mysqli_connect("localhost", "root", "") ;
  // On se place sur la base de données
  mysqli_select_db($connexion, "db_gest_wissen") ;
  // On insère un utilisateur
  $success = mysqli_query($connexion, "
    INSERT INTO Utilisateur (email, mot_de_passe)
    VALUES('matt.damon@outlook.fr', 'georgette')
  ") ;
  // Si l'utilisateur a bien été enregistré on affiche un petit message de succès
  if($success){
    echo "L'utilisateur matt.damon@outlook.fr a bien été enregistré <br/>" ;
  }
  // On récupère l'utilisateur au moyen d'une requête
  $lignes = mysqli_query($connexion, "
    SELECT *
    FROM Utilisateur
    WHERE email = 'matt.damon@outlook.fr'
  ") ;
  // Si la requête a échoué
  if(!$lignes){
    echo "La requête SQL est incorrecte. <br/>" ;
  }
  // On récupère le nombre de lignes trouvées
  $nb = mysqli_num_rows($lignes) ;
  // Si l'on a aucune ligne, c'est que l'utilisateur n'existe pas
  if($nb == 0){
    echo "L'utilisateur n'existe pas ! <br/>" ;
  }
  // On récupère la première ligne
  $ligne = mysqli_fetch_assoc($lignes) ;
  // On affiche les informations de la ligne
  echo "Utilisateur : " . $ligne["email"]  . "<br/>" ;
  echo "Mot de passe : " . $ligne["mot_de_passe"] . "<br/>" ;
?>
```

Questions :

**1.** Tout d'abord, créer la base de données « db_gest_wissen » contenant la table « Utilisateur » et les champs « email » et « mot_de_passe ».

**2.** Insérer deux utilisateurs, puis écrire la requête permettant de récupérer le 2ème utilisateur en fonction de son adresse e-mail.

**3.** Dans le dossier « tests », créer le fichier « fifthpage.php » et testons l'exemple de code ci-avant présenté.

**4.** Adapter le code précédent afin de récupérer l'utilisateur dont l'adresse email est passée au moyen du paramètre GET « email ». En déduire les adaptations à effectuer sur le fichier « connexion.php » (répertoire « wissen ») afin que l'on puisse se connecter à l'espace client au moyen des comptes utilisateurs stockés dans la base de données « db_gest_wissen ».

**5.** On souhaiterait stocker plus d'informations concernant chacun des utilisateurs. En particulier, l'on souhaiterait, pour chaque utilisateur, connaître : sa civilité, son prénom, son nom, son adresse, son code postal, sa ville, ses numéros de téléphone fixe et mobile. Adapter la base de données en conséquence, à savoir créer les champs : civilite, prenom, nom, adresse, code_postal, ville, fixe et mobile.

**6.** Rédiger la requête SQL permettant de récupérer les informations relatives à l'utilisateur ayant l'adresse email « stromae@pikachu.fr ».

**7.** Adapter le code de la page « mes_informations.php » afin de récupérer et d'afficher les informations personnelles de l'utilisateur authentifié.

## Partie 4 - Afficher des données

Dans la partie 3, nous avons récupéré une ligne stockée en base de données et en avons affiché le contenu. Cette ligne correspondait à un utilisateur. Cependant, on peut être amené à vouloir afficher le résultat d'une requête contenant plusieurs lignes. Pour ce faire, il nous faut être en mesure de parcourir les lignes retournées et en afficher le contenu. Cette opération est répétitive. Si bien qu'apparaît la notion de boucle, encore appelée structure itérative.

Notions :

**Structure alternative**

Une structure alternative ou structure conditionnelle permet d'exécuter une série d'instructions (=opérations) sous réserve qu'une condition soit vérifiée.

○ Exemple :

```php
<?php
  // on récupère l'heure actuelle grâce à la function "date"
  $date = date("H:i:s");
  if($date < "12:00:00"){
    echo "bonjour !" ;
  }else if($date < "17:00:00"){
    echo "bon après-midi !";
  }else{
    echo "bonsoir ! ";
  }
?>
```

○ Principaux opérateurs de comparaison en PHP :
- $A == $B : opérateur « égal à », soit ici A égal à B ;
- $A != $B : opérateur « différent de », soit ici A différent de B ;
- !$A : opérateur « non », soit ici non A ;
- $A && $B : opérateur « et », soit ici A et B ;
- $A || $B : opérateur « ou », soit ici A ou B.

**Fonction**

Une fonction est un fragment de code réutilisable. Une fonction peut retourner un résultat. Lorsque la fonction ne retourne pas de résultat, on parle de procédure. En PHP, on déclare une fonction grâce au mot-clef function et l'on utilise le mot-clef return pour retourner un résultat. Par ailleurs, il possible de passer des informations à une fonction en utilisant ce qu'on appelle des paramètres.

*Exemple n°1 : fonction permettant de se connecter à la base « db_gest_wissen ».*

```php
<?php
  function se_connecter(){
    // récupération d'une connexion au SGBD
    $connexion = mysqli_connect("localhost", "root", "") ;
    // sélection de la base de données « db_gest_wissen »
    mysqli_select_db($connexion, "db_gest_wissen") ;
    // retourne la connexion à la base de données « db_gest_wissen »
    return $connexion ;
  }
?>
```

*Exemple n°2 : fonction retournant un utilisateur à partir de son adresse email ou faux si l'utilisateur n'existe pas.*

```php
<?php
  function recup_utilisateur($email){
    // Connection à la base « db_gest_wissen » grâce à la fonction précédente
    $connexion = se_connecter() ;
    // Récupération de l'utilisateur dont l'email est fourni en paramètre
    $lignes = mysqli_query($connexion, "
      SELECT *
      FROM Utilisateur
      WHERE email = 'matt.damon@outlook.fr'
    ") ;
    // Récupération du nombre de lignes trouvées
    $nb = mysqli_num_rows($lignes) ;
    // Si l'on a aucune ligne, c'est que l'utilisateur n'existe pas, on retourne faux
    if($nb == 0){
      return false ;
    }
    // Récupération de la première ligne, c'est-à-dire de l'utilisateur
    $ligne = mysqli_fetch_assoc($lignes) ;
    // On retourne l'utilisateur
    return $ligne ;
  }
?>
```

*Exemple n°3 : utilisation de la fonction recup_utilisateur.*

```php
<?php
  // Récupération de l'utilisateur dont l'adresse email est matt.damon@outlook.fr
  $utilisateur = recup_utilisateur("matt.damon@outlook.fr") ;
  // Affichage du prénom et du nom de l'utilisateur
  echo "Bonjour " . $utilisateur["prenom"] . " " . $utilisateur["nom"] ;
?>
```

**Structures itératives**

Une structure itérative permet d'itérer, c'est-à-dire de répéter une série d'instructions (=opérations) plusieurs fois. On parle plus simplement de boucle.

*N.B. : afin de bien comprendre les exemples suivants, il convient préalablement de bien comprendre ce que fait la fonction mysqli_fetch_assoc. Cette fonction retourne la ligne courant et passe à la ligne suivante. Cette fonction retourne faux (false) quand il n'y a plus de lignes.*

*Exemple de boucle « POUR » : une boucle « POUR » est idéale pour répéter un traitement un nombre de fois connu à l'avance. Dans l'exemple qui suit, on affiche la liste des utilisateurs.*

```php
<?php
    // Connection à la base « db_gest_wissen »
    $connexion = se_connecter() ;
    // Récupération de tous les utilisateurs
    $utilisateurs = mysqli_query($connexion, "SELECT * FROM Utilisateur") ;
    // Récupération du nombre d'utilisateurs trouvés
```

```
    $nb = mysqli_num_rows($utilisateurs) ;
    // Parcours des utilisateurs
    for($i=0 ; $i<$nb ; $i++){
       // Récupération de la ligne courante et passage à la ligne suivante
      $utilisateur = mysqli_fetch_assoc($utilisateurs) ;
      // Affichage du prénom et du nom de l'utilisateur puis saut de ligne
      echo $utilisateur["prenom"] . " " . $utilisateur["nom"] . "<br/>" ;
    }
?>
```

*Exemple de boucle « TANT QUE » : une boucle « TANT QUE » est idéale pour répéter un traitement un nombre de fois inconnu à l'avance. Dans l'exemple qui suit, on affiche la liste des utilisateurs mais avec une boucle « TANT QUE » cette fois-ci.*

```
<?php
  // Connection à la base « db_gest_wissen »
  $connexion = se_connecter() ;
  // Récupération de tous les utilisateurs
  $utilisateurs = mysqli_query($connexion, "SELECT * FROM Utilisateur") ;
  // Récupération du premier utilisateur
  $utilisateur = mysqli_fetch_assoc($utilisateurs) ;
  // Parcours des utilisateurs tant qu'il y en a
  while($utilisateur){
    // Affichage du prénom et du nom de l'utilisateur puis saut de ligne
    echo $utilisateur["prenom"] . " " . $utilisateur["nom"] . "<br/>" ;
    // Récupération de l'utilisateur suivant
    $utilisateur = mysqli_fetch_assoc($utilisateurs) ;
  }
?>
```

Questions :

**1.** Dans le dossier « tests », créer les fichiers « sixthpage.php » et tester les précédents exemples.
**2.** Créer une page « seventhpage.php », puis y placer la procédure se_connecter() ainsi que la fonction recup_utilisateur($email).
**3.** Créer la procédure afficher_utilisateur($utilisateur) qui affiche les informations relatives à un utilisateur sur une ligne (une ligne de tableau). Tester votre procédure sur un utilisateur quelconque.
**4.** Créer la fonction recup_utilisateurs() retournant la liste des utilisateurs.
**5.** Afficher tous les utilisateurs sous forme tabulaire (tableau).

Question de gestion :

En 1 à 2 pages maximum, à l'aide de vos connaissances ainsi qu'en vous inspirer de la présente situation de gestion ou d'autres situations gestion, vous répondrez à la question suivante : en quoi un espace client, ou plus généralement un extranet, peut-il permettre d'améliorer l'efficacité et la qualité de service d'une organisation ?