UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence 2 Informatique
INFO0303 – 2020/2021
***Resp. :* C. Rabat**

Travaux dirigés n° 2

Programmation orientée objets en PHP

Exercice 1 (Actualités)
Nous souhaitons écrire une page d'un site contenant des actualités. Une actualité est caractérisée par un titre, un message et une date. Nous supposons que les actualités sont stockées dans un fichier texte. Chaque ligne contient les informations d'une actualité séparées par un " ;" (nous considérons que " ;" ne peut être présent dans le message).

1. Écrivez une classe `Actualite` contenant les attributs nécessaires, un constructeur, des getters et une méthode d'affichage (conversion en chaîne de caractères).
2. Proposez une méthode `fromString` qui permet de créer une actualité depuis une chaîne de caractères (que l'on suppose valide) contenant les données d'une actualité séparées par un " ;".
3. Donnez le script `liste.php` permettant d'afficher toutes les actualités du fichier.

Nous avons à notre disposition un script nommé `vue.php` qui crée un affichage d'une instance de `Actualite` contenue dans une variable `$actualite`.

4. Que doit-on changer dans le script `liste.php` pour utiliser ce script ?
5. Donnez le contenu du script `vue.php`.

Exercice 2 (Utilisateur)
Nous souhaitons écrire une classe `Utilisateur` permettant de gérer un formulaire de connexion. Un utilisateur possède uniquement un login et un mot de passe.

1. Donnez les attributs de cette classe, ainsi que le constructeur.
2. Écrivez la méthode `afficherForm` qui affiche les champs nécessaires dans un formulaire pour la saisie des données d'un utilisateur. Le champ de saisie du login possède comme valeur par défaut, la valeur de l'attribut login.
3. Écrivez la méthode `fromForm` qui retourne un objet `Utilisateur` à l'aide des informations saisies depuis un formulaire.
4. Écrivez la méthode `verification` qui teste si l'utilisateur a saisi des informations correctes (le login et le mot de passe sont spécifiés en constantes de la classe, vivement la BD !!!).
5. Écrivez un script qui affiche soit un formulaire de connexion, soit un message de bienvenue si l'utilisateur s'est connecté avec succès.

Exercice 3 (Classe outil)
L'ensemble des scripts CSS utilisés dans la page doit être déclaré dans des balises `meta` dans l'en-tête HTML. De même, il est conseillé de charger les scripts *Javascript* à la fin du document HTML (comme la bibliothèque *JQuery*, par exemple). De plus, du code *Javascript* doit être parfois exécuté lorsque le chargement de la page est terminé (et après l'inclusion des bibliothèques).

Pour simplifier, nous souhaitons écrire une classe `WebPage` permettant de spécifier les scripts *CSS* et *Javascript* à charger à l'aide des méthodes `addCSSScript` et `addJSScript` et le code *Javascript* à exécuter une fois la page chargée à l'aide d'une méthode `addOnlineScript`. Ces méthodes peuvent être appelées plusieurs fois et à différents endroits. Cette classe contient en plus deux méthodes `entete` et `piedDePage` qui permettent respectivement d'afficher l'en-tête de la page et le pied-de-page.

1. Comment gérer l'ajout d'un nouveau script ? Comment les stocker ?
2. Donnez le code complet de la classe `WebPage`.
3. Proposez une page contenant un *tooltip* utilisé dans *Bootstrap* en utilisant cette classe.

Pour éviter les rejeux de formulaires, nous souhaitons proposer une solution dans notre classe `WebPage`. Pour cela, nous souhaitons ajouter une méthode `getCle` qui retourne une chaîne de caractères unique qui sera ajoutée dans les différents formulaires comme champ caché, ainsi qu'une méthode `verifierCle` qui vérifie que la clé est correcte.

4. Rappelez le problème du rejeu de formulaires.
5. Comment utiliser le champ caché pour vérifier la validité d'un formulaire ?
6. Expliquez comment pourraient fonctionner les méthodes `getCle` et `verifierCle`.

Annexes - Manuel PHP

file (version simplifiée) :

```
array file ( string $filename )
```

Lit le fichier et renvoie le résultat dans un tableau (chaque ligne étant dans une case).

Utilisation des *cards* de *bootstrap* :

```
<div class="card">
  <div class="card-header">Titre</div>
  <div class="card-body">
    Contenu qui peut tout contenir... ou presque !
  </div>
  <div class="card-footer"></div>
</div>
```

Il est possible d'ajouter des classes spécifiques dans les attributs `class` des éléments comme la couleur de fond (`bg-danger`, `bg-primary`, `bg-light`...), la couleur du texte (`text-white`) ou l'alignement du texte (`text-align`).

Utilisation des *tooltips* (info-bulles) avec *bootstrap* :

Il faut ajouter les attributs spécifiques `data-toggle` et `data-placement` à la balise (`i` ou `a`, par exemple). Le texte affiché dans la bulle lorsque la souris passe sur l'élément est spécifié dans l'attribut `title`. Exemple d'une bulle affichée au-dessus de l'élément :

```
<a href="#" data-toggle="tooltip" data-placement="top" title="Texte">infos</a>
```

L'utilisation des *tooltips* nécessite d'inclure les scripts *Javascript JQuery.js* et *bootstrap.bundle.min.js*. Une fois la page chargée, il faut activer les *tooltips* avec ce code *JQuery* :

```
$(function () {
    $('[data-toggle="tooltip"]').tooltip();
}
```