L3 Informatique - 2024/2025 UE Développement Web David Lesaint

## CC PHP: Session 1 - 1h30 - Sur machine

Téléchargez l'archive cc-19-php-sujet.tgz de l'espace Moodle. Après extraction, renommez le répertoire obtenu nomprenom où nom et prenom sont vos nom et prénom écrits en minuscules et sans espaces (par ex. delafontaine-jean). A l'issue de l'examen, archivez ce répertoire (tar czf nom-prenom.tgz nom-prenom) et déposez l'archive sur Moodle.

Les exercices portent sur la mise en place d'une correction automatique en PHP de formulaires HTML dans un contexte d'examen sur machine. Des étudiants doivent développer un formulaire dans un fichier HTML sur la base d'un énoncé (non présenté ici). Le formulaire n'inclut que des champs texte et des groupes de cases à cocher let il doit être soumis au script correcteur.php. Ce dernier doit alors produire un rapport d'erreurs avec une note et afficher les notes minimum et maximum qui ont été enregistrées en base de données. La note attribuée à l'étudiant est la somme des points obtenus pour chacun des champs texte et groupes de cases divisée par le nombre total de champs et de groupes. Le barême est le suivant :

- pour chaque champ texte : 0 point si correcteur.php ne l'a pas reçu, 1 point sinon quelle que soit la valeur reçue;
- pour chaque groupe de cases : x points où x est la fraction du nombre de cases cochées qui ont été reçues parmi celles devant être pré-cochées avec l'attribut checked, les autres cases étant ignorées.

Pour établir la correction, **correcteur.php** s'appuie sur une spécification formelle du formulaire fournie dans le fichier **formulaire.json**. Les figures 1, 2, 3 et 4 illustrent un exemple de spécification au format JSON, le code source d'un formulaire totalement conforme à cette spécification, la page web obtenue au chargement, et la correction obtenue à sa soumission. Pour cette même spécification, les figures 5 et 6 illustrent un formulaire comportant des erreurs et la correction produite. Dans ce dernier exemple (voir le fichier source HTML), le second champ texte et le second groupe de cases sont mal nommés (crochets oubliés dans name="classes") et aucune case précochée du premier groupe n'a été cochée d'où la note de 0.25=(1+0+0/2+0/1)/4.

La spécification du formulaire explicite simplement les attendus sous la forme d'un objet JSON ayant 3 propriétés :

- "method" : la propriété à valeur dans {"get", "post"} <sup>2</sup> qui indique la méthode HTTP attendue pour soumettre le formulaire;
- "textFields": le tableau (potentiellement vide) d'objets qui décrivent chacun un champ texte;
- "checkboxFields": le tableau (potentiellement vide) d'objets qui décrivent chacun un groupe de cases.

Les propriétés des objets correspondant aux champs texte sont le type HTML du champ ("type"), le libellé HTML à utiliser (label), et son nom HTML ("name"). Les propriétés des objets correspondant aux groupes de cases sont le type HTML des cases ("type"), le paragraphe HTML utilisé pour la question introductive ("p"), le nom HTML des cases sans les crochets ("name"), un objet "values" dont chaque propriété donne la valeur HTML d'une case et le libellé HTML associé, et enfin le tableau des valeurs HTML des cases devant être précochées ("checked").

L'objectif est ici de compléter les trois fichiers **connexpdo.inc.php**, **correcteur.php** et **gabarit.php**. Les autres fichiers CSS et HTML sont fournis pour la mise en forme et les tests.

<sup>1.</sup> On entend par groupe de cases à cocher un ensemble de cases dont le nom est identique et se termine par [] (par ex. name="classes[]"). On permet ainsi un choix multiple et les cases cochées sont récupérables sous la forme d'un tableau PHP dont les éléments sont leurs valeurs HTML.

<sup>2.</sup> En minuscules!

Exercice 1. Créez sous phpMyAdmin une base de données MySQL de nom l3info\_cc\_19\_php\_correcteur en choisissant le moteur InnoDB et le jeu d'interclassement utf8mb4\_bin. Importez y le script l3info\_cc\_19\_php\_correcteur.sql L'import crée la table NOTES et plusieurs enregistrements.

NOTES est dotée des colonnes suivantes :

- ID : clé primaire auto-incrémentée
- FORMULAIRE: hash MD5 d'un fichier JSON de spécification de formulaire (varbinary (32)).
- ETUDIANT: nom de l'étudiant (varchar (20)).
- NOTE : note de l'étudiant (nombre décimal comportant 3 chiffres au maximum avec une précision de 2 chiffres après la virgule (decimal (3, 2))).

**Exercice 2.** Adaptez le script **connexpdo.inc.php** afin d'utiliser le compte root qui vous est attribué par le conteneur Docker pour se connecter au SGBD MySQL.

Exercice 3. Le script correcteur.php contient une classe Correcteur que l'on instancie pour produire la correction en invoquant 2 méthodes. lire\_spec est la première méthode invoquée. Elle utilise la fonction json\_decode qui convertit l'objet JSON du fichier formulaire.json en un tableau PHP associatif. La conversion transforme tout objet JSON rencontré en un tableau associatif en faisant correspondre propriété d'objet (son nom et sa valeur) et élement du tableau (sa clé et sa valeur). Complétez simplement la méthode lire\_spec pour initialiser la propriété privée \$method.

Exercice 4. La méthode vérifier est la seconde méthode invoquée. Elle appelle vérifier\_champs\_textuels et vérifier\_cases qui corrigent les champs et les groupes de cases. Ces méthodes construisent la correction et la stockent dans la propriété privée \$correction sous la forme d'un tableau associatif qui sera utilisé ensuite pour l'affichage. On donne ci-dessous la valeur du tableau \$correction correspondant à la correction de formulaire-063.html et qui illustre la structure à respecter :

```
Array (
    [textFields] => Array (
             [0] \Rightarrow Array (
                       [label] => Votre patronyme
                      [value] => Dupont
                      [points] => 1 )
             [1] => Array (
                      [label] => Votre prénom
                      [value] =>
                      [points] => 0 ) )
    [checkboxFields] => Array (
             [0] \Rightarrow Array (
                      [p] => Vos paradimes de programmation préférés
                      [score] => 1 sur 2
                      [points] => 0.5 )
             [1] \Rightarrow Array (
                      [p] => Vos classes de complexité préférées
                      [score] \Rightarrow 1 sur 1
                      [points] => 1 ) )
```

Complétez les méthodes vérifier\_champs\_textuels et vérifier\_cases à cet effet.

**Exercice 5.** En plus de \$correction, les méthodes vérifier\_champs\_textuels et vérifier\_cases mettent à jour la propriété privée \$points qui est le tableau indicé des points calculés (une valeur par champ/groupe). Complétez les méthodes vérifier\_champs\_textuels et vérifier\_cases à cet effet.

Exercice 6. La méthode vérifier appelle ensuite enregistrer\_note qui calcule la note finale, la stocke dans la propriété privée \$note, puis enregistre le résultat dans la base de données. La valeur à insérer pour la colonne FORMULAIRE est le résultat de l'application de la fonction hash\_file avec l'algorithme "md5" au fichier formulaire.json. La valeur à enregistrer pour la colonne ETUDIANT est toujours la même et égale à "Ann Onyme". La valeur à enregistrer pour la colonne NOTE doit respecter le type MySQL imposé. Complétez la méthode enregistrer\_note à cet effet.

**Exercice 7.** La méthode vérifier se conclut en appelant afficher\_résultats. Cette dernière extrait de la base de données toutes les notes obtenues pour le formulaire considéré. Elle importe ensuite le gabarit gabarit.php pour générer la page HTML affichée. Complétez la méthode afficher\_résultats à cet effet.

Exercice 8. Le gabarit gabarit.php affiche deux tableaux de résultats : un pour les champs texte, un pour les groupes de cases à cocher. Il affiche ensuite la note finale et les notes minimum et maximum enregistrées en base de données pour ce formulaire. Le gabarit a accès à ces données via les propriétés \$correction et \$note de l'objet Correcteur et via le tableau de notes extrait de la base de données par la méthode afficher\_résultats. Complétez le fichier gabarit.php pour produire le résultat escompté en respectant les consignes suivantes :

- utilisez la fonction array\_walk pour générer le tableau des cases à cocher;
- utilisez les classes CSS de gabarit.css pour colorier en vert (resp., rouge, orange) les lignes des tableaux ayant 1 point (resp., 0 point, p points avec  $p \in ]0,1[$ ).

Exercice 9. On souhaite afficher dans la console Javascript du navigateur 3 le nombre de fois où un même nom a été saisi dans le champ dénommé patronyme. On affichera par exemple Dupont : 5 tentatives. On comptabilisera aussi les cas où aucun nom n'aura été reçu ou communiqué en associant la soumission au nom "Ann Onyme", par exemple Ann Onyme : 2 tentatives. Compléter correcteur.php (sans rajouter de propriétés ou de méthodes à la classe) et gabarit.php à cet effet.

Exercice 10. On souhaite maintenant intégrer à notre application la correction de formulaires contenant des boutons radio. Le dossier formulaire-bis contient deux exemples de formulaires HTML pour une même spécification : un qui est correct (formulaire-1.html) et un qui ne l'est pas (formulaire-07.html)). Le formulaire attendu étend le formulaire des questions précédentes en y ajoutant deux boutons radio. Copiez vos fichiers formulaire.json, correcteur.php et gabarit.php dans formulaire-bis et adaptez-les en conséquence. Le barême est de 1 point si l'un des deux boutons a été coché, 0 points sinon (l'attribut checked est ignoré), et un tableau doit être affiché pour les boutons radio tel qu'illustré en figures 7 et 8.

<sup>3.</sup> Le code HTML <script>console.log("text")</script>y affichera la chaîne text.

```
1⊝ {
             "method": "post",
            "textFields": [
 40
                          type": "text".
                         "label": "Votre patronyme",
                         "name": "patronyme"
 8
 9⊜
                         "type": "text",
"label": "Votre prénom",
"name": "prénom"
11
12
13
                 }
15⊜
             "checkboxFields": [
16⊜
                  {
                         "type": "checkbox",
"p": "Vos paradimes de programmation préférés",
17
                        "name": "paradigmes",

"values": {
    "f": "Fonctionnel",
    "l": "Logique",
    "o": "Objet"
19
20 €
21
22
23
24
25⊜
                          checked": [
                              "f",
"l"
27
28
29
                        1
                  },
30⊝
                         "type": "checkbox",
"p": "Vos classes de complexité préférées",
31
32
                         "name": "classes",

"values": {

"P": "La classe P",

"NP": "La classe NP"
33
34⊜
35
36
37
38⊜
                          checked": [
39
                               "NP'
10
                        ٦
                 }
42
           ]
43 }
```

FIGURE 1 - formulaire.json

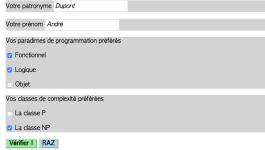


FIGURE 3 – formulaire-1.html



FIGURE 5 - formulaire-025.html



FIGURE 2 – Source formulaire-1.html

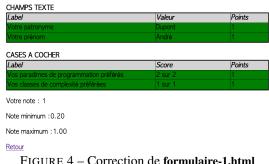


FIGURE 4 – Correction de formulaire-1.html

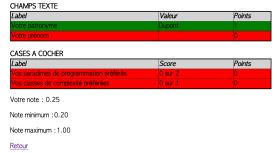
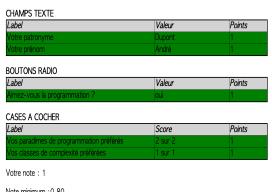


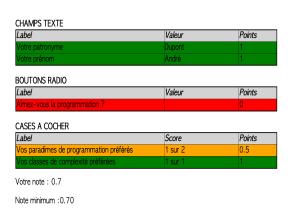
FIGURE 6 – Correction de formulaire-025.html



Note minimum : 0.80 Note maximum : 1.00

Retour

FIGURE 7 – Correction de formulaire-1.html



Retour
FIGURE 8 – Correction de formulaire-07.html

Note maximum: 1.00