

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 **nom-prenom** 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 cocher¹ et 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 barème 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"}`² 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.

1. 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.

2. 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'encodage `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 élément 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] => Array (
            [label] => Votre patronyme
            [value] => Dupont
            [points] => 1 )
        [1] => Array (
            [label] => Votre prénom
            [value] =>
            [points] => 0 ) )
    [checkboxFields] => Array (
        [0] => Array (
            [p] => Vos paradigmes de programmation préférés
            [score] => 1 sur 2
            [points] => 0.5 )
        [1] => Array (
            [p] => Vos classes de complexité préférées
            [score] => 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 navigateur³ 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 barème 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.

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

```
1 {
2   "method": "post",
3   "textFields": [
4     {
5       "type": "text",
6       "label": "Votre patronyme",
7       "name": "patronyme"
8     },
9     {
10      "type": "text",
11      "label": "Votre prénom",
12      "name": "prénom"
13    }
14  ],
15  "checkboxFields": [
16    {
17      "type": "checkbox",
18      "p": "Vos paradimes de programmation préférés",
19      "name": "paradigmes",
20      "values": {
21        "f": "Fonctionnel",
22        "l": "Logique",
23        "o": "Objet"
24      },
25      "checked": [
26        "f",
27        "l"
28      ],
29    },
30    {
31      "type": "checkbox",
32      "p": "Vos classes de complexité préférées",
33      "name": "classes",
34      "values": {
35        "P": "La classe P",
36        "NP": "La classe NP"
37      },
38      "checked": [
39        "NP"
40      ]
41    }
42  ]
43 }
```

FIGURE 1 – formulaire.json

```
9
10 <body>
11 <!-- Formulaire -->
12 <form method="post" action="correcteur.php">
13   <p>
14     <label>Votre patronyme</label>
15     <input type="text" name="patronyme" value="Dupont" />
16   </p>
17   <p>
18     <label>Votre prénom</label>
19     <input type="text" name="prénom" value="André" />
20   </p>
21   <div>
22     <p>Vos paradimes de programmation préférés</p>
23     <p>
24       <input type="checkbox" name="paradigmes[]" value="f" checked />
25       <label>Fonctionnel</label>
26     </p>
27     <p>
28       <input type="checkbox" name="paradigmes[]" value="l" checked />
29       <label>Logique</label>
30     </p>
31     <p>
32       <input type="checkbox" name="paradigmes[]" value="o" />
33       <label>Objet</label>
34     </p>
35   </div>
36   <div>
37     <p>Vos classes de complexité préférées</p>
38     <p>
39       <input type="checkbox" name="classes[]" value="P" />
40       <label>La classe P</label>
41     </p>
42     <p>
43       <input type="checkbox" name="classes[]" value="NP" checked />
44       <label>La classe NP</label>
45     </p>
46   </div>
47   <input type="submit" value="Vérifier !" />
48   <input type="reset" value="RAZ" />
49 </form>
50 </body>
51 </html>
```

FIGURE 2 – Source formulaire-1.html

Votre patronyme Dupont

Votre prénom André

Vos paradimes de programmation préférés

☒ Fonctionnel

☒ Logique

☐ Objet

Vos classes de complexité préférées

☐ La classe P

☒ La classe NP

Vérifier !

RAZ

FIGURE 3 – formulaire-1.html

CHAMPS TEXTE		
Label	Valeur	Points
Votre patronyme	Dupont	1
Votre prénom	André	1

CASES A COCHER		
Label	Score	Points
Vos paradimes de programmation préférés	2 sur 2	1
Vos classes de complexité préférées	1 sur 1	1

Votre note : 1

Note minimum : 0.20

Note maximum : 1.00

[Retour](#)

FIGURE 4 – Correction de formulaire-1.html

Votre patronyme Dupont

Votre prénom

Vos paradimes de programmation préférés

☐ Fonctionnel

☐ Logique

☒ Objet

Vos classes de complexité préférées

☐ La classe P

☒ La classe NP

Vérifier !

RAZ

FIGURE 5 – formulaire-025.html

CHAMPS TEXTE		
Label	Valeur	Points
Votre patronyme	Dupont	1
Votre prénom		0

CASES A COCHER		
Label	Score	Points
Vos paradimes de programmation préférés	0 sur 2	0
Vos classes de complexité préférées	0 sur 1	0

Votre note : 0.25

Note minimum : 0.20

Note maximum : 1.00

[Retour](#)

FIGURE 6 – Correction de formulaire-025.html

CHAMPS TEXTE		
Label	Valeur	Points
Votre patronyme	Dupont	1
Votre prénom	André	1
BOUTONS RADIO		
Label	Valeur	Points
Aimez-vous la programmation ?	oui	1
CASES A COCHER		
Label	Score	Points
Vos paradigmes de programmation préférés	2 sur 2	1
Vos classes de complexité préférées	1 sur 1	1
Votre note : 1		
Note minimum : 0.80		
Note maximum : 1.00		
Retour		

FIGURE 7 – Correction de **formulaire-1.html**

CHAMPS TEXTE		
Label	Valeur	Points
Votre patronyme	Dupont	1
Votre prénom	André	1
BOUTONS RADIO		
Label	Valeur	Points
Aimez-vous la programmation ?		0
CASES A COCHER		
Label	Score	Points
Vos paradigmes de programmation préférés	1 sur 2	0.5
Vos classes de complexité préférées	1 sur 1	1
Votre note : 0.7		
Note minimum : 0.70		
Note maximum : 1.00		
Retour		

FIGURE 8 – Correction de **formulaire-07.html**