



M3104 – Programmation Web côté serveur (2018/19)

TDm 4 / TDm 5

Le code source de votre application, un export de votre base de données et la documentation technique de votre application seront à rendre sur la plateforme **UMTice**. L'application finalisée sera également mise à disposition sur la plateforme <http://la-myweb.univ-lemans.fr/> dont l'URL précise figurera dans la documentation.

NB : ce TD est prévu sur 2 séances. Les fichiers sont à déposer **au plus tard le vendredi 5 octobre à 13h** sur UMTice.

Au cours de ces 2 séances, nous allons mettre en place un mécanisme permettant de mesurer et d'analyser la fréquentation d'un site Web. L'idée est de créer un composant objet qui sera instancié à chaque chargement d'une page et mettra à jour les statistiques d'accès à cette page (gérées dans une base de données MySQL).

Question 1 : spécifiez et implémentez une classe **Statistiques** que vous utiliserez pour comptabiliser les accès à chaque script dans lequel cette classe sera instanciée. La mise à jour du nombre d'accès se fera naturellement dans le constructeur de la classe. La classe *Statistiques* contient également une méthode **showPageAccess()** permettant de retourner le nombre d'accès comptabilisés pour le script en cours d'exécution. Testez votre classe en l'instanciant dans différents contextes.

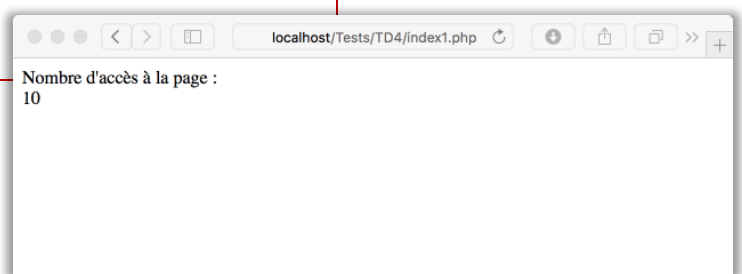
Lors du premier appel à partir d'un script donné, le compteur correspondant à cette page est ajouté dans la base de données et initialisé à '1'.

Vous pourrez avoir besoin d'utiliser la variable d'environnement `PHP_SELF` (issue du tableau associatif `$_SERVER` dans lequel de nombreuses variables d'environnement du serveur sont disponibles) qui donne le nom du script en cours d'exécution (on y accède de la façon suivante : `$_SERVER['PHP_SELF']`).


Exemple :

```
<?php
    require('Statistiques.class.php');

    $stats = new Statistiques();
    echo "Nombre d'accès &agrave; la page : <br />";
    echo $stats->showPageAccess();
?>
```



Question 2 : Ajoutez une méthode `getStats()` retournant (sous forme d'un tableau HTML) les statistiques d'accès à l'ensemble des pages du site. Testez votre méthode en instanciant la classe ainsi modifiée.



Page	Nombre access
/Tests/TD4/index1.php	12

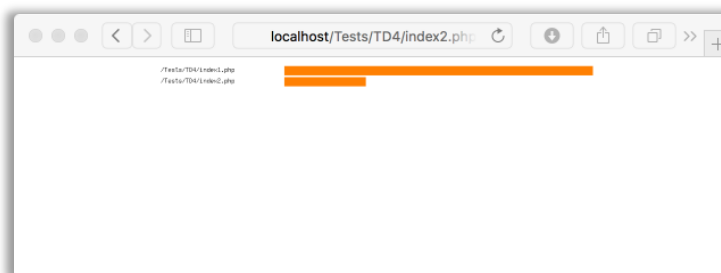
Question 3 : On souhaite désormais améliorer le mécanisme de calcul de la fréquentation afin de comptabiliser le nombre de machines différentes ayant accédé à une page donnée (une machine étant considérée dans un premier temps comme étant identifiée par son adresse IP, *i.e.* la variable d'environnement `REMOTE_ADDR` présente dans `$_SERVER`). Testez et instanciez votre nouvelle classe `getStatsByIP()`.



On 2017-09-19 :

IP	Page	Nombre access
127.0.0.1	/Tests/TD4/index1.php	1
::1	/Tests/TD4/index1.php	17

Question 4 : On cherche ensuite à générer automatiquement une image PNG représentant les statistiques de fréquentation (plutôt qu'un tableau HTML). Implémentez la méthode `getStatsPNG()` permettant d'envoyer une image au format PNG dans le flux de sortie.



L'idée de base du script que vous devez écrire sera donc d'interroger la base de données dans le but de récupérer les résultats, de construire l'image à l'aide de ces résultats et de finalement l'envoyer au client. L'image contiendra, pour chaque page présente dans la base, un histogramme (un rectangle) proportionnel à sa fréquentation ainsi qu'un rappel de son nom. L'histogramme correspondant à la page la plus fréquentée occupera la totalité de l'espace alloué au dessin des histogrammes, les autres seront proportionnels à cet histogramme.

Vous disposez désormais d'un composant objet pouvant être instancié à chaque chargement d'une page et mettant à jour les statistiques d'accès à cette page (gérées dans une base de données MySQL). Une difficulté tient néanmoins au fait que vous identifiez les visiteurs par leur adresse IP, ce qui ne permet pas toujours de bien discriminer des visiteurs uniques (par exemple s'ils sont derrière un proxy).

Question 5 : On cherche donc à améliorer notre mécanisme d'identification des utilisateurs afin de mieux analyser les pratiques individuelles. Proposez et implémentez un mécanisme basé sur l'utilisation de cookies afin d'identifier le navigateur d'un utilisateur et non plus seulement son adresse IP. Modifiez en conséquence la méthode *getStats()* afin qu'elle affiche le nombre d'accès unique (navigateurs uniques) pour chaque page.

Question 6 : La méthode précédente permet d'identifier le navigateur, mais pas la personne qui l'utilise. Proposez et implémentez une évolution de votre mécanisme basé sur des cookies afin d'identifier (et de comptabiliser) les pages visitées par les utilisateurs authentifiés sur le site web (il faudra donc, en particulier, implémenter un mécanisme d'authentification dans l'esprit de ce que vous aviez fait au TDm2).

Question 7 : On souhaite alors modifier la méthode *getStats()* afin de pouvoir afficher des statistiques de fréquentation journalières. Après avoir, si besoin, modifié votre modèle de données, il vous est demandé de concevoir et d'implémenter dans une méthode *getDailyStats()* le mécanisme permettant de segmenter les statistiques jour par jour (par utilisateur unique).

Question 8 : On souhaite enfin améliorer la méthode *getStatsPNG()* afin qu'elle permette soit la création d'une image PNG globale comme à la question 4, soit une image PNG restreinte à la fréquentation d'un jour donné transmis en paramètre. Concevez et implémentez *getDailyStatsPNG()*, cette variation de *getStatsPNG()*.

Question bonus : On souhaite pouvoir exporter les statistiques de fréquentation du site (toutes ou seulement celles d'une date en particulier) sous la forme d'un flux XML. Concevez, implémentez et testez une méthode *exportInXML()* d'export en utilisant l'extension *SimpleXML*.