


EXERCICES D'APPLICATION SUR LES OBJETS PHP et MYSQL

Essayez toujours dans un premier temps de trouver la réponse par vous même sur le web. Demander la solution à son voisin ne vous aides malheureusement pas pour le futur. Ni de le regarder faire. Si vous êtes bloqué relisez le cours adéquat.

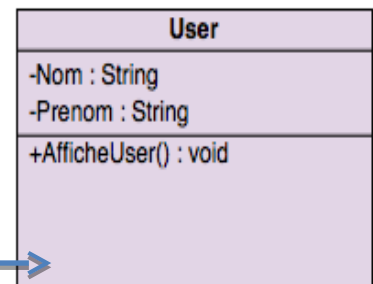
Continuer d'utiliser la Page de garde de votre espace web !

| | |
|--|---|
| <p>Afin de pouvoir facilement valider vos compétences en web dynamique PHP et MYSQL :</p> <p>proposer une page de garde de votre site qui donne accès à l'ensemble de vos exercices.</p> | <div><div>Menu</div><div><div>Mon Site D'exercices TP</div></div><div><div>Exo 1 Php</div><div>Exo 2 php</div><div>Exo 3 BDD</div><div>Exo 4 Etc</div><div>Exo 5</div><div>Exo 6</div><div>...</div></div></div> |
|--|---|

Exercice 1. (création d'une classe en php)

Vous devez faire cet exercice en suivant le cours (ne pas copier coller le code de vos anciens exercices !) le code source doit être disponible dans votre github dans un repertoire PhpObjet. Le code source doit être visible sur votre page dans une balise : `<pre></pre>`

- 1) Créer une classe User qui possède 2 propriétés : Nom et Prenom ainsi qu'une methode afficheUser qui echo « je suis un User »
- 2) Créer 1 users dans votre page index.html puis appeler sa méthode afficheUser
- 3) En disign CSS HTML reproduire le schémas de classe de votre objet : exemple :



Exercice 2. (création d'un constructeur)

Une classe peu posséder une méthode spéciale : « constructeur ». Cette méthode est automatiquement appelée lorsqu'un objet de cette classe est instancié (créée)

- 1) Créer une classe Personnage qui possède 2 propriétés : Speudo et Vie ainsi qu'un constructeur vide : `public function __construct(){}`
- 2) Implémenter le constructeur pour qu'il initialise la propriétés vie à 100

Exercice 3. (création d'un constructeur avec paramètre)

Une classe peu posséder une méthode spéciale : « constructeur ». Comme c'est une méthode elle peu recevoir des paramètres. **Attention à partir du moment ou il y a un constructeur qui attend des paramètres vous ne pouvez plus instancier votre classe sans paramètre.**

- 1) Reprendre la classe Personnage et implémenter le constructeur pour qu'il initialise la propriétés vie à 100 et initialise la propriété pseudo avec la valeur passer en paramètre
- 2) Créer un personnage Exemple `personnage1 = new personnage("Julien") ;`
- 3) Ajouter une méthode dans votre classe pour afficher le nombre de point de vie du personnage ainsi que son pseudo.

Exercice 4. (interaction entre objet)

Les méthodes des classes peuvent prendre des paramètres (\$variables) qui peuvent être aussi des Objets

- 1) créer une méthode "attaquer" qui prend en paramètre un autre personnage
- 2) créer une méthode " défense" qui prend en paramètre une valeur d'attaque.
- 3) Implémenter la méthode attaquer : elle doit appeler la méthode défense de l'objet personnage passé en paramètre et lui mettre 50 de dégat
- 4) Implémenter la méthode défense pour que la valeur d'attaque passée en paramètre décrémente la propriété vie du personnage.
- 5) Rajouter des méthodes et des echos pour afficher les etapes de jeu.
- 6) Instancier 2 personnage dont l'un attaque l'autre.
- 7) Afficher le résultat de déroulement de l'attaque en HTML.

Exercice 5. (objet PDO Select)

- 1) Créer en BDD une table personnage avec les propriétés de la classe personnage. (Pensez toujours à rajouter un champ id)
- 2) Créer plusieurs Personnage avec phpMyadmin dans votre Table
- 3) Modifier votre constructeur de Classe pour avoir en paramètre un id
- 4) Dans votre classe Personnage instancier un objet PDO dans la méthode constructeur puis faites un `select * from personnage where id = id .` (je vous conseil de mettre l'objet PDO dans une propriété de votre classe)
- 5) Initialiser les propriétés de votre Classe Personnage avec les données récupéré de la requête BDD
- 6) Instancier un Personnage et afficher ses informations. Exemple : `$P1 = new Personnage(12) ; $P1->AfficherPersonnage() ;`

Exercice 6. (objet PDO Update)

- 1) Dans la méthode défense, utiliser le PDO de votre Personnage pour faire une requête update pour modifier les points de vie de votre personnage en BDD
- 2) Vérifier en base que vos personnages perdent bien leur vie après un combat (**pour faire un combat c'est Personnage1->attaque(Personnage2) ;**)
- 3) Faites une méthode soin pour remettre la vie du personnage à son maximum et le mettre à jour dans la BDD

Exercice 7. (objet PDO Select ALL)

- 1) Créer plusieurs personnage dans votre BDD.
- 2) en PHP faites une requête select * from Personnage avec un objet PDO instancier dans votre program principal (pas dans la classe). faites une boucle while sur les résultats et instancier chaque objet Personnage en le rangeant dans un tableau.

rappel: \$Tableau = Array() ;

Ajouter une case : `array_push($tableau, $Personnage);`

Récupérer un objet personnage : `$Personnage = $Tableau[indice] ;`

- 3) parcourez en fin de page votre tableau (foreach) récupérer l'objet Personnage dedans et afficher son pseudo et ses points de vie avec la méthode afficherPersonnage de votre classe Personnage

Exercice 7. (objet PDO Insert)

- 1) créer un formulaire HTML qui possède les champs représentant les propriétés de votre Classe Personnage.
- 2) En validant le formulaire, ajouter un Personnage en BDD.
- 3) Reproduisait le même mécanisme mais le PDO et L'ajout en BDD doivent être positionné dans une méthode de votre classe.

Exercice 8. (objet PDO delete)

- 1) créer un formulaire HTML qui possède un input select avec tous les pseudos de votre Jeu. (utiliser select * from Personnage) , la value de chaque option doit être ID du Personnage.
- 2) En validant le formulaire, supprimer le Personnage sélectionné en BDD.
- 3) Reproduisait le même mécanisme mais le PDO et L'ajout en BDD doivent être positionné dans une méthode de votre classe.

Mettez à disposition votre repository github pour M. Langlacé