CT JS/PHP : Session 2 - Durée 2h - Sur machine

Téléchargez l'archive **ct-22-2.tgz** déposée sur l'espace Moodle puis décompressez-la dans votre répertoire personnel. Placez-vous dans le répertoire créé (**ct-22-2**) qui contient différents fichiers à compléter. A l'issue de l'examen, compressez ce répertoire et déposez votre archive sur Moodle.

On se propose de développer un mini-site web permettant de consulter et d'enrichir une liste de villes. La liste répertorie des villes, leurs régions et leurs populations.

Les 3 exercices **peuvent être traités indépendamment, le dernier comptant pour la moitié de la note.** Pour visualiser ce qui est attendu, un démonstrateur est à votre disposition.

Exercice 1. (**PHP : Tableaux/Fichier/Génération HTML**)
Cet exercice porte sur la manipulation du tableau `$villes` (fourni par l'inclusion du fichier **src/data.php**), la création de fichier et la génération HTML dans le but de créer un tableau permettant de modifier le nombre d'habitants des préfectures (cf. Figure 1). Répondez aux 4 questions de cet exercice au sein du fichier **tableau-villes.php**.

1. Filtrez `$villes` pour ne garder que les préfectures.
2. Triez `$villes` dans l'ordre décroissant de population.
3. Exportez `$villes` au format csv. Un exemple de fichier à obtenir vous est donné par **src/villes.csv**.
4. Générez les lignes de modification du tableau pour chaque élément de `$villes`. Une ligne (`<tr></tr>`) est composée d'une case (`<td></td>`) affichant le nom de la ville, d'une seconde affichant le nom de la région et d'une troisième contenant un champ de saisie numérique (`<input type="number" />`) qui a pour valeur la population de la ville et qui a pour nom `population[i]` où i est l'identifiant de la ville. N'hésitez pas à inspecter le démonstrateur pour comprendre l'architecture HTML du tableau.

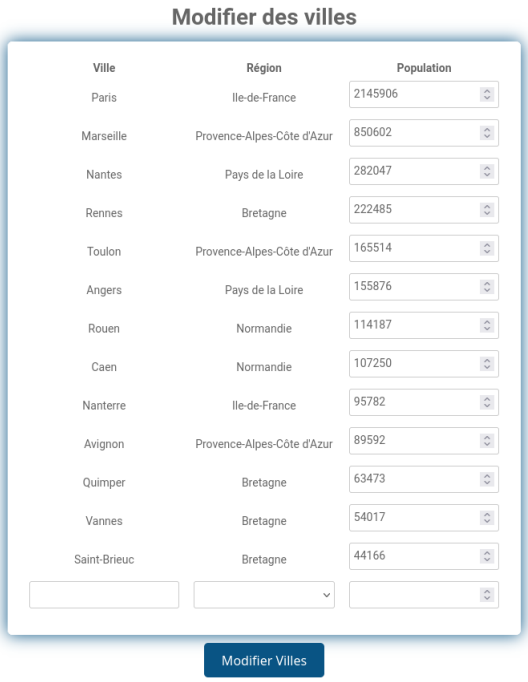

FIGURE 1 – Tableau d'ajout et de modification de villes à générer dans l'exercice 1 et à utiliser dans l'exercice 2.

Exercice 2. (**PHP : Bases de données**)
Cet exercice porte sur la manipulation en PHP de la base `l2mi_ct_22_2_population` pour modifier et insérer des villes. La structure de la base est illustrée en Figure 2. Répondez aux 2 questions de cet exercice au sein du fichier **modification-ajout-villes.php**.

Importez le fichier **src/l2mi_ct_22_2_population.sql** sous PhpMyAdmin pour créer cette base qui contient quelques enregistrements. Pensez également à adapter **src/connexpdo.inc.php** avec vos informations de connexion.
Le tableau de cette page (cf. Figure 1) est un formulaire de modification et d'ajout de ville où la dernière ligne sert à ajouter une nouvelle ville en base et les autres à modifier le nombre d'habitants de chaque ville présente en base. N'hésitez pas à inspecter la page et les valeurs envoyées par le formulaire pour faciliter la compréhension et la manipulation. Comme il s'agit de données saisies par l'utilisateur et d'opérations répétitives, la non-utilisation de requêtes préparées sera pénalisée.

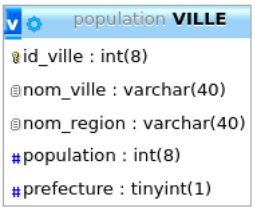

FIGURE 2 – Modèle de la base de données `l2mi_ct_22_2_population`

1. Modifiez en base les villes dont les nombres d'habitants sont renseignées.
2. Ajoutez en base la ville de la dernière ligne du tableau si tous les champs de cette ligne sont renseignés.

Exercice 3. (**JS**)
Complétez le fichier **population.js** sans modifier le fichier **population.html** pour répondre aux questions qui suivent.

1. Créez un constructeur `Ville` permettant de construire un objet ayant 3 propriétés (2 chaînes `nom` et `region` et 1 entier `population`) à partir des arguments passés en entrée. Construisez ensuite le tableau d'objets `villes` contenant un objet `Ville` pour chaque ville renseignée dans le tableau HTML (en récupérant les données par l'API DOM). Si vous ne parvenez pas à construire ce tableau d'objets, trouvez une solution de substitution afin de pouvoir répondre aux questions qui suivent.

2. Implantez la fonction `permutation_villes_regions` qui, à chaque appel, permute les deux premières colonnes du tableau HTML (voir Figure 3). Ne supprimez aucun des objets correspondants aux cellules du tableau : utilisez la méthode `insertBefore` pour cela. Veillez à permuter également les 2 en-têtes des colonnes. Utilisez aussi la variable globale `permutation` qui vous est fournie afin que la fonction puisse déterminer et mettre à jour l'état de la page (colonnes permutées ou non).
Testez la fonction en console puis implantez un écouteur qui l'appellera à chaque clic sur l'icône situé en haut à gauche du tableau.

Villes

Région	Ville	Population
Pays de la Loire	Nantes	280
Bretagne	Rennes	220
Pays de la Loire	Angers	150
Normandie	Rouen	110
Normandie	Caen	100
Bretagne	Quimper	60
Bretagne	Vannes	50
Bretagne	Saint-Brieuc	40

FIGURE 3 – Permutation $n + 1$ des 2 premières colonnes.

Villes

Ville	Région	Population
Angers	Pays de la Loire	150
Caen	Normandie	100
Nantes	Pays de la Loire	280
Quimper	Bretagne	60
Rennes	Bretagne	220
Rouen	Normandie	110
Saint-Brieuc	Bretagne	40
Vannes	Bretagne	50

FIGURE 4 – Tri en ordre croissant des noms de villes.

Villes

Région	Ville	Population
Bretagne	Vannes	50
Bretagne	Saint-Brieuc	40
Normandie	Rouen	110
Bretagne	Rennes	220
Bretagne	Quimper	60
Pays de la Loire	Nantes	280
Normandie	Caen	100
Pays de la Loire	Angers	150

FIGURE 5 – Tri en ordre décroissant des noms de villes sur colonnes permutées.

3. Implantez la fonction `tri_villes` qui, à chaque appel, trie le tableau d'objets `villes` selon un ordre sur les noms de villes : le premier tri se fait dans l'ordre croissant (voir Figure 4), le second dans l'ordre décroissant et ainsi de suite. La variable globale `tri_croissant_villes` vous est fournie afin que la fonction puisse déterminer et mettre à jour l'état de la page (tableau trié en ordre croissant ou décroissant). Testez la fonction en console.
Implantez ensuite la fonction `tri_colonnes_villes` qui, à chaque appel, (1) appelle `tri_villes` puis (2) met à jour les cellules de chaque ligne du tableau HTML à partir du tableau trié `villes`. Veillez à ce que le tri soit effectif, que les 2 premières colonnes aient été permutées ou non (voir Figure 5).
Testez la fonction en console puis implantez un écouteur qui l'appellera à chaque clic sur l'icône situé à côté de l'en-tête `Ville`.

4. Implantez un écouteur sur chaque cellule de la colonne `Population` qui, à chaque clic, incrémentera de 10 la population affichée (voir Figure 6). Tout clic sera inopérant si la population a atteint ou dépassé 300.

5. La fonction `population_moyenne` prend en argument un nom de ville et une population pour cette ville. Implantez-la afin de (1) mettre à jour l'objet correspondant dans le tableau `villes` et (2) calculer et afficher la population moyenne dans la cellule située au dessus de l'en-tête `Population` (voir Figure 7). Etendez

Villes

⇄		
Ville	Région	Population
Angers	Pays de la Loire	160
Caen	Normandie	100
Nantes	Pays de la Loire	280
Quimper	Bretagne	60
Rennes	Bretagne	220
Rouen	Normandie	110
Saint-Brieuc	Bretagne	40
Vannes	Bretagne	50

FIGURE 6 – Clic sur la population d'Angers.

Villes

⇄		127.50
Ville	Région	Population
Angers	Pays de la Loire	160
Caen	Normandie	100
Nantes	Pays de la Loire	280
Quimper	Bretagne	60
Rennes	Bretagne	220
Rouen	Normandie	110
Saint-Brieuc	Bretagne	40
Vannes	Bretagne	50

FIGURE 7 – Mise à jour de la population moyenne.

l'implémentation de l'écouteur développé en question précédente afin d'appeler la fonction à chaque fois qu'une population est incrémentée pour réactualiser la moyenne.