Semaine 6 - Technologies du Web

décembre 2020

Travail de synthèse

Ce travail prolonge la fiche n°6 et reprend également des éléments des fiches précédentes (4 et 5 en particulier). Il est demandé de constituer un projet unique appelé `synthese` (dans un dossier de même nom) pour les 3 (ou 4) exercices ci-dessous.
Les `seuls` scripts qui pourront être invoqués directement seront

- `synthese/index.php` (construit à partir de `td6/index.php` avec les modifications demandées dans les exercices).
- `synthese/createUser.php` (voir exercice 2)
- `synthese/services/*` l'ensemble des services demandés.

Vous veillerez à respecter les règles qui ont prévalu pour tous les fiches de TD : structuration du projet, regroupement des pages elles-mêmes dans un dossier `views` protégé (non accessible directement), protection des dossiers `lib`, `etc`...
Les spécifications des services devront être `rigoureusement respectées` : les noms des services et de leurs arguments ne doivent pas être modifiés. Les réponses et résultats des services doivent suivre le cahier des charges (vos services pourront être testés directement via des scripts automatisés).
Les services demandés dans la fiche 6 doivent être conservés.
Quelques aménagements de la page principale vous sont demandés dans les exercices 1 et 3. Vous pouvez en faire quelques autres si vous le souhaitez mais en conservant les principaux éléments. Vous serez certainement amené à modifier le CSS fourni (qui est minimal ;-)) et il vous est conseillé de l'améliorer nettement.

Exercice 1 : *Ajouter des critères de sélection*

Un seul critère de sélection des communes était implémenté dans la fiche 6 : la sélection selon le territoire. Nous voulons implémenter d'autres critères. Les critères se combineront par un « et » (comme dans l'exercice de la feuille 4) :

- sélection par nom : ne sélectionner que les communes dont le nom contient une sous-chaîne fixée par l'utilisateur, sans tenir compte de la « casse » (majuscules/minuscules).
- sélection par surface minimale : sélectionner uniquement les communes d'une superficie supérieure ou égale à une valeur exprimée en hectares ($1ha = 10000\ m^2$)

Q 1 . Modifier le service `getCommunes` qui pourra prendre en compte plusieurs critères de filtres. La valeur de chaque critère de filtre est indiquée au service par une variable HTTP. L'absence d'un paramètre (ou une valeur vide) indique que ce critère n'est pas utilisé. Les critères sont cumulatifs.

mode d'envoi des paramètres : GET ou POST		
`nom`	type	signification
`territoire`	entier ≥ 0	territoire d'appartenance
`nom`	chaîne	sous-chaîne du nom de commune (case insensitive)
`surface_min`	nombre	surface minimale, en hectares

Par exemple la requête `services/getCommunes.php?nom=EL&surface_min=10.5`
renvoie la liste des communes de plus de $105000m^2$ dont le nom contient `el`, sans filtrage par territoire.
Q 2 . Complétez le formulaire pour que l'utilisateur puisse fixer (ou non) ces 3 critères.
Q 3 . (facultative)
Ajouter d'autres critères de filtres. Par exemple `pop_min` (population minimale, selon le recensement de 2016) ou tout autre critère à votre convenance.

Exercice 2 : *Base : création d'utilisateur*

Vous réutiliserez la table d'utilisateurs définie pour la fiche 5, **avec mots de passe cryptés**.
Q 1 . Créez un service `services/createUser.php` qui crée un utilisateur dans la base en prenant en compte les arguments **obligatoires** suivants :

mode d'envoi des paramètres : POST uniquement		
`nom`	type	signification
`nom`	chaîne non vide	nom du nouvel utilisateur
`prenom`	chaîne non vide	prénom du nouvel utilisateur
`login`	chaîne non vide	identifiant nom du nouvel utilisateur
`password`	chaîne non vide	mot de passe

Réponse du service :

— si des paramètres sont manquants ou incorrects : `status="error"`

— si un utilisateur avec ce login existe déjà : `status="error"`

— sinon `status="ok"` `result=` login de l'utilisateur créé.

Q 2 . Créez à la racine du projet un script `register.php` qui affiche une page contenant

— un formulaire de création de compte permettant la saisie des 4 informations nécessaires

— une zone de message

La validation du formulaire entraîne l'affichage d'un message d'attente et l'invocation du service créé à la question précédente. À la réception de la réponse du service, la zone message indique si la création a été effectuée ou non (et dans ce cas en indiquer la raison). La page en cours ne doit pas être rechargée, à aucun moment.

Exercice 3 : ***Authentification***

Q 1 . Écrire un service **`services/login.php`** qui prend en compte les arguments **obligatoires** suivants :

mode d'envoi des paramètres : POST uniquement		
`nom`	type	signification
`login`	chaîne non vide	identifiant utilisateur
`password`	chaîne non vide	mot de passe

Réponse du service :

— si des paramètres sont manquants ou incorrects : `status="error"`

— si l'utilisateur était déjà en mode connecté : `status="error"`

— si login ou password sont incorrects : `status="error"`

— sinon `status="ok"` `result=` un objet contenant les attributs `login`, `nom`, `prenom` de l'utilisateur connecté
Par ailleurs, le service ouvre une session indiquant que l'utilisateur est connecté.

Par exemple l'invocation de l'URL `login.php` avec les paramètres

login	mallani
password	animal

renvoie une réponse dont l'attribut `result` vaut `{"login":"mallani","nom":"Annie", "prenom":"Malle"}` (en admettant que on n'était pas déjà en mode connecté)

Q 2 . Écrire un service **`services/logout.php`** sans argument.

Réponse du service :

— si l'utilisateur n'était pas connecté : `status="error"`

— sinon `status="ok"` `result=`login de l'utilisateur déconnecté. Par ailleurs, le service met fin à la session.

Q 3 . Ajoutez à la page principale du site une zone qui contient

— pour un utilisateur non connecté : un bouton (ou équivalent) de connexion. L'utilisateur qui l'active peut ensuite entrer son login et mot de passe et se connecter

— pour un utilisateur connecté : affiche ses nom, prénom et login, ainsi qu'un bouton de déconnexion.

La connexion et la déconnexion ne doivent pas entraîner de rechargement de la page.

Exercice 4 : *Facultatif (bonus)*

Il s'agit de permettre a un utilisateur connecté (donc authentifié) d'avoir une liste de « villes favorites ». Pour un utilisateur connecté la liste de ses villes favorites est affichée, en plus des autres informations. L'utilisateur doit pouvoir ajouter ou supprimer une ville favorite.

Le dispositif de sélection et d'annulation de ville favorite est laissé à votre choix. Vous pourrez par exemple prévoir dans la liste des villes obtenues par recherche et dans la liste des favoris un bouton à côté de chaque commune permettant de l'ajouter ou la supprimer comme favorite (selon le cas).

Pour cette question il vous faudra réaliser 3 services. Chacun des services fonctionnera uniquement en mode connecté (si une session attestant d'une connextion n'est pas en cours, le status du service doit être "error")

`services/getFavoris.php`
Aucun paramètre
`result`
Liste des communes favorites de l'utilisateur connecté. Même structure que pour getCommunes.

`services/addFavori.php`		
mode d'envoi des paramètres : GET ou POST		
`nom`	type	signification
`insee`	chaîne non vide	commune à ajouter aux favoris
`result`		
Liste contenant seulement la commune ajoutée. Même structure que précédemment		

`services/removeFavori.php`		
mode d'envoi des paramètres : GET ou POST		
`nom`	type	signification
`insee`	chaîne non vide	commune à retirer des favoris
`result`		
code INSEE de la commune retirée		