Sujet 9 - Technologies du Web

mars 2020

Services et pages dynamiques

Au préalable, terminer l'exercice 2 de la feuille précédente

Une arborescence de fichiers vous est fournie (notez qu'elle contient de nombreux éléments de correction des exercices de la feuille précédente). Il s'agira de concevoir une page permettant d'afficher plusieurs informations différentes à propos de la course cycliste, de façon indépendante et sans recharger la page. La page est accessible par le script `multi.php` qui inclut `views/pageMulti.php`. Cette page devrait afficher 3 informations :

- les informations concernant une équipe à choisir
- le tableau d'arrivée des coureurs une étape à choisir
- les statistiques générales de la course

Ces informations étaient déjà demandés dans la fiche précédente, sur des pages séparées. À chaque type d'information correspond une méthode dans la classe `DataLayer`.

Mise en place

Installez les fichiers fournis sur le serveur, ajoutez votre propre fichier `lib/db_parms.php` et ouvrez la page `multi.php` dans le navigateur.
Vous constaterez que, pour l'instant, seules les statistiques sont affichées (mais le bouton de mise à jour est inactif). L'envoi des formulaires (« équipe » ou « arrivée ») ne produit pas non plus le résultat attendu.

L'objectif de l'exercice est de construire les services et d'écrire les fichiers javascript nécessaires au fonctionnement des différentes fonctionnalités.

Exercice 1 :
Tous les web-services renverront un objet JSON ayant pour structure :
`status :` vaut `"ok"` ou `"error"`
`message :` est défini si status vaut error : message d'erreur
`result :` est défini si status vaut ok. Sa composition varie selon les services.

Q 1 . Au cours de cette question, nous allons mettre en place l'affichage de l'équipe grâce au formulaire `form_equipe`.

1. Partie serveur : écriture du service (script à compléter)

   services/getInfoEquipe.php fournit les informations concernant une équipe
   paramètre HTTP (GET) :
   `equipe :` (le nom d'une équipe) chaîne non vide obligatoire

   réponse : voir ci-dessus. Pour ce service, `result` est un objet ayant pour attributs :
   `nom :` chaîne
   `couleur :` chaîne
   `directeur :` chaîne

   Voici un exemple de ce que produit ce service :

   ```
   {"status":"ok",
    "result":{"nom":"PicsouBank","couleur":"Or","directeur":"Raymond"}
   }
   ```

   Plusieurs cas d'erreurs sont à prévoir :

   - la requête HTTP est incomplète (argument `equipe` non fourni)
   - l'interrogation de la couche de données a été impossible (PDOException s'est déclenchée)

— l'équipe n'existe pas (la méthode `getInfoEquipe` a renvoyé false)

`status` ne doit donc valoir ok qu'en cas d'absence de toute erreur.

Vous testerez directement le service en entrant l'URL dans le navigateur, avec une valeur pour `equipe`. N'oubliez pas de tester le comportement dans les divers cas d'erreur.

2. Partie client (navigateur) : interrogation du service
   Le script `action_equipe.js` vous est partiellement fourni , il est à compléter.
   — la fonction `sendFormEquipe()` vous est fournie. Elle provoque l'envoi des données du formulaire au service `services/getInfoEquipe.php`. Vous constaterez que le résultat est traité par une fonction `processAnswer()`. Puis, en cas de succès, le résultat de `processAnswer()` est traité par `displayInfoEquipe()`
   — écrivez la fonction `processAnswer()` (4 lignes) qui reçoit en argument l'objet renvoyé par le serveur. Si `answer.status` vaut ok , son résultat est `answer.result`. Sinon elle déclenche une Error avec pour message `answer.message`
   — écrivez la fonction `displayInfoEquipe()` (8 lignes) qui reçoit en argument les infos sur une équipe (avec ses 3 attributs nom, couleur, directeur) et affiche ces informations dans le div de la classe résultat

Q 2 . Écrivez de même le service

services/getArrivee.php fournit les informations concernant une équipe **paramètre HTTP (GET) :** `etape` **:** (numéro d'étape) : entier obligatoire	**réponse :** l'attribut `result` est le résultat de la méthode `getArrivee()` de la couche données.

Testez le service.

Écrivez un javascript `action_etape.js` permettant d'activer le deuxième formulaire (NB : certaines fonctions js de la question précédente peuvent être réutilisées telles quelles, ne les réécrivez pas).

Q 3 . Faites de même un service

services/getStats.php fournit des statistiques sur la course **paramètre HTTP (GET) :** aucun	**réponse :** l'attribut `result` est le résultat de la méthode `getStats()` de la couche données.

puis un javascript `action_stats.js` de façon à activer le bouton de mise à jour.

Q 4 . Écrivez le service

services/getCoureurs.php fournit les informations concernant une équipe **paramètre HTTP (GET) :** `equipe` **:** (nom d'équipe) : chaîne obligatoire	**réponse :** l'attribut `result` est le tableau des coureurs de l'équipe

Complétez le javascript `action_equipe.js` pour afficher la liste des coureurs à la suite du descriptif de l'équipe (mais toujours dans l'élément `div` de la classe `résultat`).
Pour éviter d'aller chercher une liste des coureurs d'une équipe qui n'existerait pas, vous lancerez l'appel au service `services/getCoureurs.php` uniquement à la réception des infos sur l'équipe (donc quand l'existence de l'équipe est assurée).